UNIVERSITATEA TEHNICĂ „GH. ASACHI“ IAŞI
FACULTATEA DE AUTOMATICĂ
ŞI CALCULATOARE

PROIECT DE DIPLOMA

Coordonator:
Şef Lucr. Dr. Ing. SILION RADU

Absolvent:
IOAN I. ALEODOR DANIEL

- 2003 -

Universitatea Tehnică "Gheorghe Asachi" – Iași

Facultatea de Automatică și Calculatoare

Proiect de diplomă

Proiectarea, realizarea și testarea unui sistem de dezvoltare cu microprocesor Z80

COORDONATOR:

ș. l. dr. ing. Silion Radu

ABSOLVENT:

Ioan I. Aleodor Daniel

– 2003 –

Cuprins

1. Introducere. ..... 3.

1.1. Scopul proiectului. Utilitatea unui sistem de dezvoltare cu microprocesor. De ce Z80 ?

1.2. Schema bloc a sistemului.

2. Proiectarea sistemului. Descrierea blocurilor funcționale. ..... 7.

2.1. Microprocesorul Z80. Selecția memoriei și a dispozitivelor periferice.

2.2. Blocul generator de tact și obținerea frecvențelor pentru interfața serială.

2.3. Blocul numărător – temporizator programabil și controler de întreruperi.

2.4. Interfața serială - RS 232.

2.5. Porturile paralele.

2.6. Interfața I2C realizată hardware.

2.7. Conectorul pentru LCD compatibil Seiko.

2.8. Blocul de comutare a alimentării bancului – RAM1 pe baterie.

2.9. Posibilități de extensie a sistemului. Transformarea lui în calculator.

3. Realizarea și testarea sistemului. ..... 51.

3.1. Proiectarea și realizarea cablajului.

3.2. Tehnici de testare a sistemelor cu microprocesor.

3.3. Testarea și punerea la punct a sistemului de dezvoltare.

4. Program de transfer pe port paralel cu PC-ul. ..... 61.

5. Concluzii. ..... 72.

Anexe. ..... 75.

Capitolul 1.

Introducere.

1.1. Scopul proiectului. Utilitatea unui sistem de dezvoltare cu microprocesor. De ce Z80 ?

Scopul principal al proiectului este demonstrarea anumitor cunoștințe, metode, tehnici și abilități specifice, necesare în proiectarea și realizarea sistemelor cu microprocesor, aplicate în cazul particular al microprocesorului **Z80**, dar cu valențe suficient de generale. Orientarea proiectului este în general către hardware, urmărind detalii de proiectare, dar într-o permanentă corelație cu posibilitățile eventuale de utilizare și cu resursele software disponibile, deoarece la realizarea oricărui sistem programabil aceste aspecte sunt inseparabile.

În al doilea rând, am avut în vedere proiectarea unui sistem de dezvoltare (nucleu de dezvoltare hardware / software) cât mai universal, prevăzut cu resursele necesare pentru a putea fi folosit într-o gamă cât mai largă de aplicații, de la testarea unor echipamente electronice sau hardware, la dezvoltarea și punerea la punct de programe pentru microsisteme dedicate, la aplicații concrete de tip control automat în timp real (calculator de proces) și până la extinderea și transformarea sa în microcalculator (prin adăugarea unui modul video și a unei interfețe de disc, în acest caz sistemul având rol de "placă de bază"), fără a insista totuși pe un exemplu concret de utilizare, care ar fi făcut obiectul altui proiect.

Nu neapărat în ultimul rând, proiectul are și un anumit rol didactic, putând fi folosit pentru exemplificarea și înțelegerea problemelor generale de proiectare a unui sistem de calcul cu microprocesor.

Un ***sistem de dezvoltare*** este un nucleu de calcul care posedă bogate resurse hardware și software, realizat în jurul unui microprocesor sau microcontroller, fiind util proiectanților în realizarea de sisteme dedicate pentru o anumită aplicație. Microsistemele specializate vor avea, de regulă, o amploare mai restrânsă decât sistemul de dezvoltare. Utilitatea unui astfel de sistem universal rezultă din caracteristicile specifice proiectării aplicațiilor cu microprocesoare sau microcontrollere.

În primul rând, acesta constituie baza de plecare în familiarizarea cu tehnicile de programare specifice unui anumit tip de microprocesor (microcontroller), având un rol esențial în etapa de "training" a proiectanților, datorită în special resurselor extinse disponibile în sistem (memorie **RAM** de dimensiune suficientă pentru încărcarea programelor de pe un calculator

gazdă, monitoare software pentru vizualizarea și modificarea anumitor resurse, cu posibilități suplimentare de asamblare / dezasamblare locală a unor porțiuni mici de cod și simulare prin trasare pas-cu-pas a programelor).

Sistemul de dezvoltare va fi de mare folos și în proiectarea hardware prin faptul că el pune la dispoziție nucleul gata realizat (același în toate microsistemele dedicate, indiferent de aplicație), până la nivelul interfețelor tipizate. Proiectantul hardware va realiza, deci, modulul specializat pe care-l va conecta la sistemul de dezvoltare, având astfel posibilitatea de a-l experimenta și pune la punct, chiar dacă nu are de la început întreg sistemul realizat.

În faza de simulare a programelor, sistemul de dezvoltare reprezintă instrumentul ideal, deoarece execută în timp real și fără nici o interpretare setul de instrucțiuni al microprocesorului. În acest fel simularea este completă, pentru că se poate testa nu numai corectitudinea semantică a programelor, ci și timpul de execuție al programelor de aplicație în diferitele situații reale ce apar în exploatare. Validarea corectitudinii întregului proiect se va face cu ajutorul sistemului de dezvoltare la care s-a cuplat hardware-ul specializat, rulând programele care implementează aplicația dată, îndeplinind astfel și ultima fază care apare în proiectarea unei structuri cu microprocesor.

Chiar dacă nu mai este de mare actualitate, am ales un sistem cu microprocesor în locul unui sistem cu microcontroller datorită în primul rând complexității sale și a diverselor aspecte tehnice pe care le atinge un astfel de proiect. Proiectarea unui sistem de dezvoltare cu microcontroller (de exemplu, cu **I8051**) atinge prea puține aspecte de proiectare hardware, deoarece majoritatea resurselor sunt disponibile deja pe cip într-o versiune minimală, fiind necesară doar extinderea lor. Un alt motiv ar fi acela că deja am proiectat și realizat personal un sistem de dezvoltare cu microcontrollerul **I80C32**, în cadrul disciplinei "*Proiectarea cu microprocesoare II*".

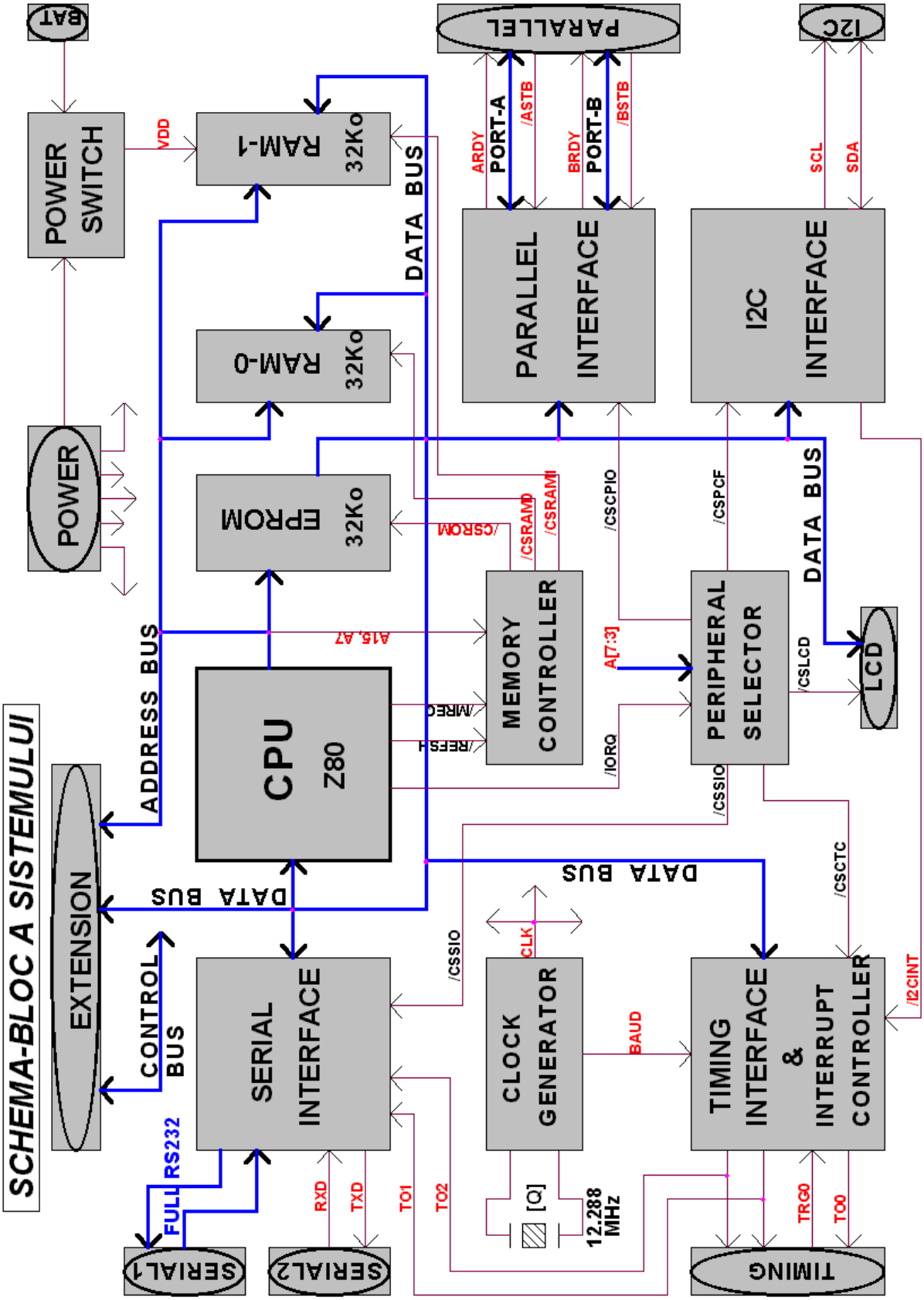
Atunci, de ce tocmai cu Z80 ? Există trei motive principale. În primul rând, proiectarea unui sistem de calcul în jurul unui microprocesor mai avansat, de exemplu **I8086**, ar fi fost prea complicată, necesitând prezența multor circuite auxiliare și a unei cantități mari de memorie pentru a beneficia de avantajele unui astfel de microprocesor, iar *realizarea sa practică nu se justifică, un sistem cu **I8086** imitând aproape în întregime structura unei plăci de bază a unui calculator **IBM-PC***. Tot în acest sens, aș aminti că acest microprocesor este mai puțin bun pentru aplicații de control, având un set de instrucțiuni orientat mai mult spre realizarea unor sisteme mari de calcul.

În al doilea rând, am încercat să scot astfel în evidență faptul că *microprocesorul Z80 nu este chiar atât de "depășit"*, fiind încă soluția optimă în cazul unor sisteme de tip calculator de proces industrial pentru aplicații complexe de conducere automată, în care abordarea distribuită

folosind mai multe microcontrollere conectate în rețea este ineficientă, datorită creșterii exagerate a complexității comunicațiilor (programele de comunicație capătă o pondere mai mare chiar decât aplicația propriu-zisă) și a costului ansamblului.

Ultimul motiv este *o oarecare experiență de tip "hobby"* în lucrul cu microcalculatoarele "**TIM-S**", "**COBRA**" și "**HC-91**", bazate pe un astfel de microprocesor, la care am realizat o serie de modificări și extensii hardware pentru adăugarea unor module adiționale (extinderea memoriei **RAM**, interfață paralelă, interfață serială, interfață analogică, programator **EPROM**-uri), precum și o serie de subrutine (modificări în memoria **ROM** sau în sistemul de operare **CP/M**) și programe de sistem (pentru gestiunea fișierelor de pe dischetă și transfer de date cu **PC**-ul).

1.2. Schema bloc a sistemului:



Capitolul 2.

Proiectarea sistemului. Descrierea blocurilor funcționale.

2.1. Microprocesorul Z80. Selecția memoriei și a dispozitivelor periferice.

Microprocesorul **Z80** (alias **Z8400**) este o unitate centrală de procesare pe un singur cip, organizată în jurul unei magistrale interne de date cu lățimea de 8 biți și constituie nucleul unei familii de circuite integrate **LSI** performante, care permit realizarea unui sistem de calcul folosind un număr redus de componente. Produsă inițial de firma "Zilog" în tehnologie **NMOS**, familia de circuite **Z80** se mai găsește încă în fabricație la diverse firme sub forma unor versiuni îmbunătățite atât tehnologic (**CHMOS** – **Z84C00**, frecvențe de tact mai mari – până la **33 MHz**) cât și constructiv (unitate centrală cu oscilator și memorie pe aceeași capsulă compatibilă la pini – **Z84C50**, periferice integrate pe un singur cip – **Z80KIO**, inclusiv în versiune cu tot cu unitatea centrală – **Z84C15**, dezvoltări arhitecturale compatibile ca set de instrucțiuni – **Z80180**, unele chiar de mare actualitate și putere de calcul – **eZ80**).

Structura internă a circuitului **Z80-CPU** (**fig. 2.1.1.**) **[20.]** are ca elemente principale unitatea aritmetico-logică – **ALU** cu registrele temporare asociate (**TEMP** și **ACU**), blocul de registre generale și secvențiatorul de control, care conține în plus registrele hardware (**I** = registrul de pagină pentru tabela de întreruperi și **R** = contorul de reâmprospătare a memoriilor dinamice) și bistabilii de configurare / validare a sistemului de întreruperi mascabile (**IM2-IM1**, respectiv **IFF1** și **IFF2**).

Caracteristică microprocesorului **Z80** este posibilitatea de comutare prin instrucțiuni a acumulatorului de lucru (**A**) împreună cu registrul indicatorilor de condiții (**F**) cu perechea de rezervă – **AF'** (***EX AF,AF'***) și a celor 2 bancuri formate din perechile de registre **BC**, **DE**, **HL**, un singur banc fiind activ la un moment dat (***EXX***). Microprocesorul execută și operații aritmetice pe 16 biți (cu două treceri prin **ALU**), folosind drept acumulator registrul dublu **HL**, sau registrele index **IX** și **IY**, care permit în plus și adresarea indexată a unor tabele în memorie. "Acumulatorul" de 16 biți – **HL** poate fi și el schimbat cu registrul **DE** prin comanda unui multiplexor cu instrucțiunea ***EX DE,HL***.

Este interesant de amintit că registrele index "de 16 biți" (teoretic, acestea se pot folosi numai pe 16 biți) **IX** și **IY**, pot fi folosite și pe 8 biți utilizând așa-numitele "instrucțiuni ascunse" (nedeclarate de firmă), prin redirectarea instrucțiunilor care folosesc regiștrii **H** și **L** în mod

separat, cu ajutorul prefixelor pentru **IX** (**DDh**) și **IY** (**FDh**), fără a folosi un asamblor special cu suport pentru instrucțiuni ascunse **[1.]**.

Dimensiunea spațiului de memorie direct adresabil este de 64Ko, contorul program – **PC** și indicatorul de stivă – **SP** fiind pe 16 biți, iar spațiul de intrare/ieșire se poate adresa direct pe 8 biți (prin acumulator), sau indirect pe 16 biți, cu perechea **BC**.

Numărul mare de registre interne a permis utilizarea mai puțin convențională în unele aplicații a microprocesorului **Z80**, fără meorie **RAM**, numai cu **EPROM**.

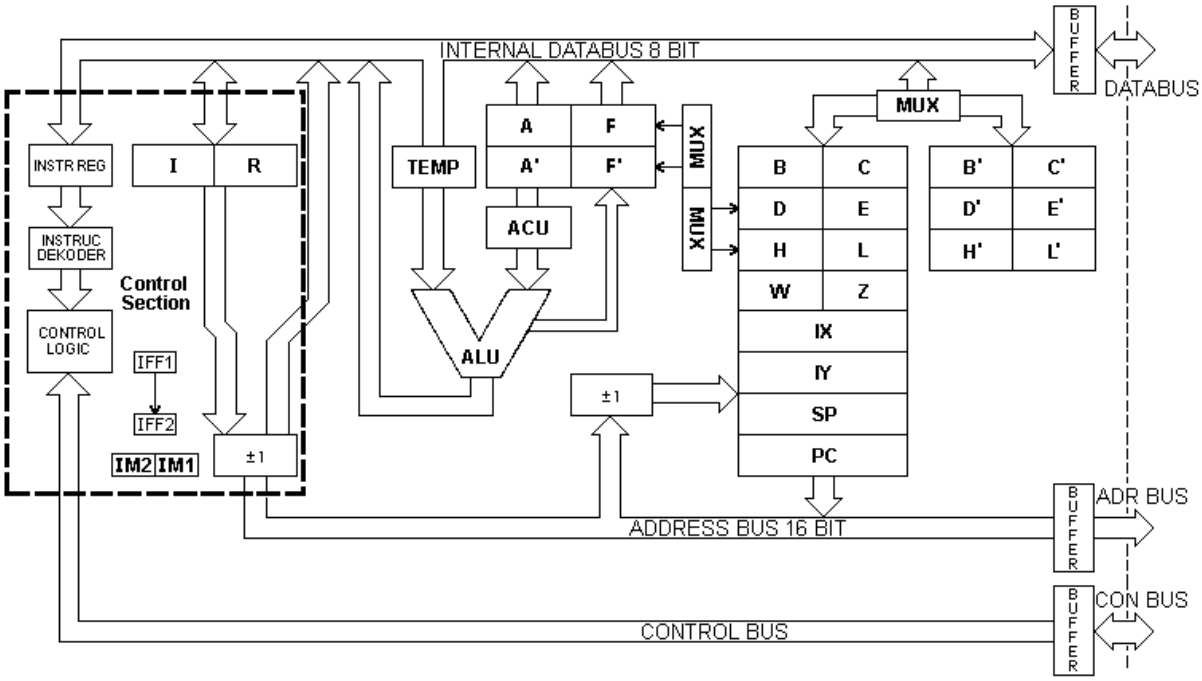


Fig. 2.1.1.

Din punct de vedere hardware, prin utilizarea unui număr redus de componente auxiliare se poate realiza un sistem puternic, cu complexitatea între sistemele cu microcontroller și sistemele de tip calculator **[3.]**. Microprocesorul **Z80** nu necesită controller de magistrale, latch-uri pentru demultiplexare sau amplificatoare de magistrale, permițând în plus conectarea ușoară a memoriilor dinamice (fără acces dublu pentru reâmprospătare). Prin folosirea circuitelor periferice specifice familiei, care conțin fiecare module interne pentru controlul sistemului de întreruperi cu propagare serie a priorităților, se poate evita și folosirea unui controller de întreruperi dedicat (lanțul de comunicație serială pentru propagarea priorităților între circuite creează un controller virtual de întreruperi distribuit).

Din punct de vedere software, microprocesorul are un set puternic de instrucțiuni **[1.]**, cu orientare generală, atât spre aplicații de control automat (***posibilitate de setare / resetare / testare***

a biților, schimbare rapidă de context prin comutarea registrelor, bucle și salturi relative, instrucțiuni de apelare / revenire condiționată din subrutine) cât și spre realizarea de microcalculatoare **[5.]** (***diversitatea modurilor de adresare și lățimea de bandă mare a setului de instrucțiuni pe memoria externă conduce la lucrul comod cu variabile și structuri de date de mari dimensiuni, indicatorul de stivă pe 16 biți permite programarea algoritmilor recursivi, instrucțiunile complexe de transfer și căutare pe blocuri permit lucrul fără DMA***).

Ar mai fi de amintit aici și cele 3 variante de lucru cu întreruperi, care acoperă atât moduri simple, utile în aplicațiile de control cât și modul specific vectorizat (cu adresare dublu indirectă și posibilitate de plasare a tabelei de întreruperi în orice zonă de adrese), mai performant chiar decât la **I8086**.

Toate aceste caracteristici hardware și software amintite situează microprocesorul **Z80** undeva între microcontrollere și microprocesoarele mai performante, având putere de calcul mai mare decât microcontrollerul **I8051** și o flexibilitate mai bună decât a microprocesorului **I8086**. Se poate spune că **Z80** înglobează principalele avantaje din ambele tipuri de circuite, putându-le înlocui cu succes într-o gamă largă de aplicații. Din aceste motive, **Z80** nu poate fi considerat chiar complet "depășit".

Primul modul din schema sistemului proiectat se referă la conectarea microprocesorului în circuit (**fig. 2.1.2.**), cu semnalele sale specifice **[2.]**.

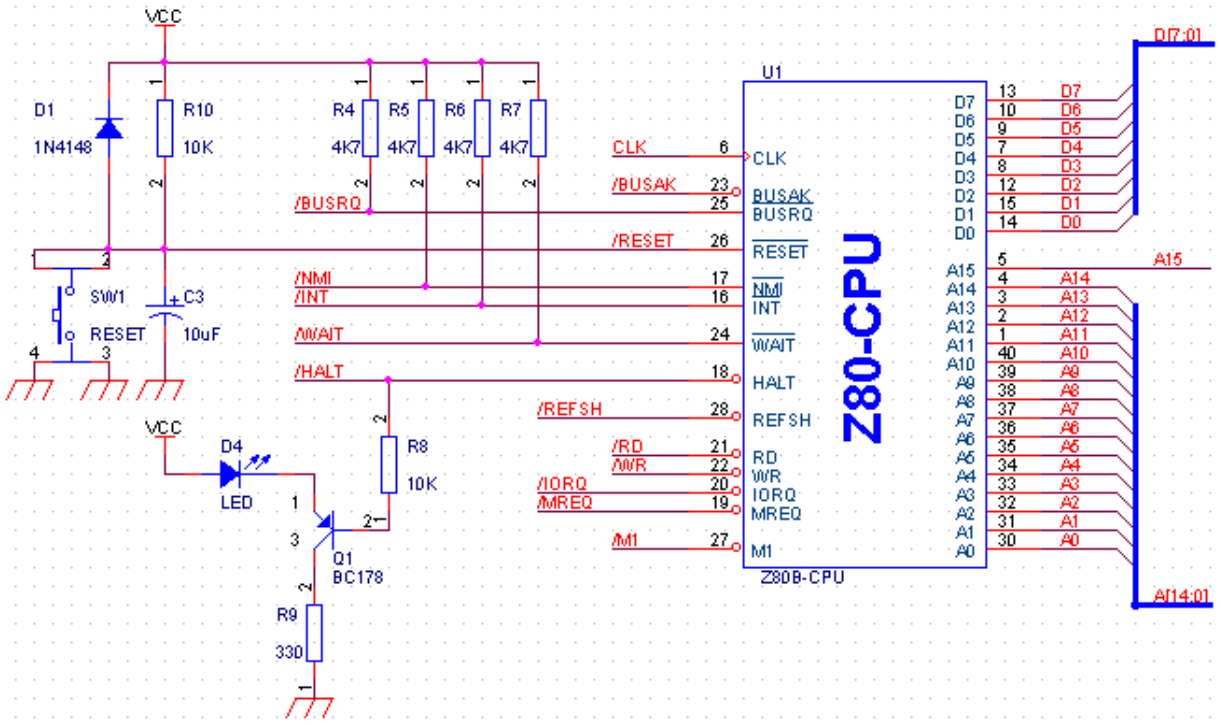


Fig. 2.1.2.

Intrările de întrerupere nemascabilă – **/NMI** și cererea de cedare a magistralelor – **/BUSRQ** nu sunt folosite în sistem, fiind invalidate prin rezistoare "pull-up" la Vcc și scoase la conectorul de extensie pentru eventuale utilizări ulterioare. De altfel, conectorul de extensie conține toate semnalele microprocesorului, fiind un conector de tip magistrală. Rezistoare cu aceleași valori (4.7K – valoare optimă) se găsesc și pe liniile **/INT** (cerere de întrerupere mascabilă) și **/WAIT** (semnal de prelungire a ciclilor), dar având un rol diferit, acela de a realiza funcții "AND cablat" cu rol de SAU pentru semnale active *LOW*, împreună cu ieșirile cu drena în gol corespunzătoare de la circuitele periferice (cererile de întreruperi, respectiv semnalele de așteptare de la cele două canale seriale).

În schemă se remarcă și prezența unui circuit clasic de resetare a sistemului la conectarea tensiunii de alimentare (D1, R10, C3) – care produce întârzierea necesară prin încărcarea condensatorului, cu posibilitate de resetare de la buton. Dioda D1 are rolul de a descărca rapid condensatorul la căderea tensiunii de alimentare.

O particularitate mai interesantă este conectarea unui indicator cu LED pe linia **/HALT** a microprocesorului. Am montat acest circuit cu rolul de a indica o posibilă eroare în sistem (după un test nereușit de memorie la inițializare, se oprește microprocesorul prin instrucțiunea ***HALT***), dar el poate fi folosit și în alte scopuri (de exemplu, în cazul unui sistem de operare în timp real se poate urmări încărcarea procesorului cu întreruperi, programul principal fiiind instrucțiunea ***HALT***). Deoarece linia **/HALT** de la **Z80** este activă *LOW*, a fost necesară conectarea unui tranzistor PNP (**BC 178**, de uz general) care să se satureze la activarea în "0" logic a semnalului. Am conectat dioda LED în emitor pentru a decala deschiderea tranzistorului în jos, spre valori mai mici a tensiunii pe bază **[8.]**:

$$Vdesc = Vcc - (Vbe + Vled) = 5V - (0.65V + 1.95V) = 2.4V\ .$$

Fără acest artificiu de schemă, dacă dioda ar fi fost conectată în colector, tranzistorul s-ar fi deschis imediat ce potențialul pe linia **/HALT** ar fi scăzut sub valoarea:

$$Vdesc = Vcc - Vbe = 5V - 0.7V = 4.3V\ .$$

Dacă se utilizează în sistem un procesor **Z80** în tehnologie **NMOS**, valorile *HIGH* ale semnalelor pe pini nu depășesc 3.6V, deci LED-ul ar fi fost aprins permanent ...

În continuare, urmează o scurtă prezentare a celor mai semnificativi cicli-mașină cu diagrame de timp **[16.]** a căror cunoaștere este esențială în proiectarea schemelor de selecție pentru circuitele de memorie și periferice, descriere importantă mai ales pentru că am folosit anumite artificii și particularități specifice microprocesorului **Z80**.

Cel mai important ciclu-mașină este ***ciclul "fetch"***, în care se face extragerea și interpretarea instrucțiunilor din memorie (**fig. 2.1.3.**). Acest ciclu este semnalat în cazul microprocesorului **Z80** de activarea semnalului specific "*Machine cycle one*" (**/M1**), urmat la

jumătate de tact de cererea de acces la memorie (**/MREQ**) și de semnalul **/RD** care validează citirea. Prezența semnalului **/M1** este utilă mai ales pentru posibilitatea de rulare pas-cu-pas a programelor, cu oprire hardware după fiecare instrucțiune prin stoparea ceasului (față de alte microprocesoare, **Z80** este static pe stare, adică nu-și modifică starea internă la oprirea ceasului pe o perioadă de timp nedefinită), dar poate fi folosit și în alte scopuri (de exemplu, la separarea memoriei program de memoria de date plus stivă, atunci când memoria program este "read only"). Linia **/MREQ** se folosește împreună cu linii de adresă la selecția cipurilor de memorie (**/CE**), iar semnalul **/RD** se conectează la pinul de validare a ieșirii pe magistrala de date (**/OE**), atât pentru circuitele de memorie **ROM** cât și pentru memoria **RAM**.

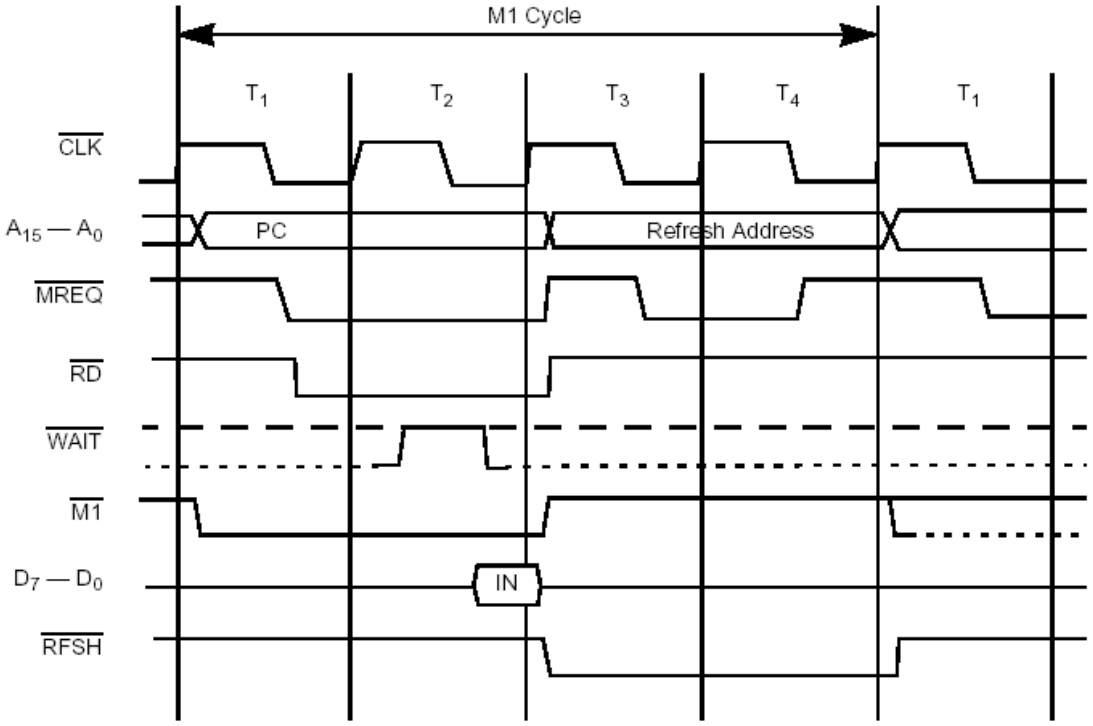


Fig. 2.1.3.

În timpul decodificării instrucțiunii, deoarece magistralele microprocesorului își pot schimba conținutul (instrucțiunea fiind deja extrasă din memorie), pe magistrala de adrese poate apărea acum conținutul perechii de registre interne **I-R**, adică registrul-pagină de întreruperi – **I** (pe partea *HIGH*) și contorul intern de reâmprospătare a memoriilor dinamice – **R** (pe partea *LOW*), care este incrementat în fiecare ciclu "fetch". Acest fapt asigură o utilizare extrem de eficientă a magistralei de adrese, prin lipsa unor "timpi morți" de inactivitate **[1.]**. Momentul schimbării informației de pe magistrală este semnalat prin activarea liniei **/REFSH** care semnalează prezența adresei de reâmprospătare. Acest semnal poate fi utilizat la schimbarea

regimului de lucru al memoriei (pregătirea **RAM**-urilor dinamice pentru "refresh"), dar nu poate fi strob efectiv pentru adresa de reâmprospătare, deoarece magistrala nu este încă stabilă în momentul apariției sale. Din acest motiv, microprocesorul mai activează încă o dată în ciclul "fetch" linia **/MREQ**, la jumătate de tact de la schimbarea adresei pe magistrală. De abia apariția frontului negativ pe **/MREQ** atunci când **/REFSH** este deja pe "0" logic poate încărca sigur adresa de "refresh" în memoria dinamică.

S-ar putea pune întrebarea de ce am insistat pe această descriere dacă la proiectarea sistemului nu am folosit memorii dinamice (în prezent fiind disponibile memorii statice de capacități suficiente) ?! Am folosit totuși aceste particularități ale microprocesorului **Z80**, la paginarea memoriei **EPROM** cu bancul de memorie **RAM0**, printr-un artificiu mai puțin obișnuit.

Deoarece contorul de reâmprospătare lucrează modulo 128, folosind numai 7 biți din registrul **R**, bitul 8 nu este afectat de mecanismul de incrementare automată, fiind disponibil pentru programator. Acest bit poate fi pus pe "0" sau pe "1" prin instrucțiunile de transfer între registrul **R** și acumulator, apariția sa pe magistrala de adrese putând fi memorată și folosită ca un bit de port, provenit direct dintr-unul din regiștrii interni ai microprocesorului.

Avantajul esențial al acestui mod de expandare, pe lângă evitarea pierderii unui bit dintr-un port **I/O**, se bazează pe o caracteristică a ciclului "fetch", și anume aceea că reâmprospătarea memoriei dinamice are loc întotdeauna după extragerea codului instrucțiunii curente. Din acest motiv, dacă bitul a fost poziționat imediat înaintea unei instrucțiuni de salt codificată pe un singur octet, schimbarea configurației memoriei se va face după extragerea instrucțiunii, microprocesorul "știind" deja din configurația anterioară unde va continua cu programul în noua configurație. Rezultatul este practic posibiliattea de salt și apel între bancuri (pagini) de memorie expandate, care sunt mapate pe aceeași zonă de adrese **[13.]**.

De exemplu, pentru salt necondiționat între bancuri, se poate folosi secvența următoare:

```
LD   HL, adresă   ; încărcare adresă de salt
LD   A, R         ; citire bit 7 din R
XOR  80H          ; complementare bit 7 în acumulator
LD   R, A         ; modificare bit 7 cu apariție pe magistrală după "fetch"-ul următor
JP   (HL)         ; salt la adresa din HL, după comutare
```

De asemenea, se poate realiza și apelul unor subrutine la adresele fixe de "restart" ***p=00H, 08H, 10H, 18H, 20H, 28H, 30H, 38H***, dintr-un banc în celălalt, prin folosirea instrucțiunii "**RST p**". Singura observație care trebuie făcută în acest caz este aceea că adresa de

revenire va fi salvată în stivă în bancul nou, după comutare, dacă stiva este alocată pe zona de adrese care se paginează. Din acest motiv, adresa de revenire trebuie preluată cu registrul **HL** din stivă înainte de comutarea înapoi, subrutinele apelate în bancul celălalt trebuind să se termine în loc de ***RET***, cu secvența:

```
POP  HL      ; se preia adresa de întoarcere din stivă, din bancul apelat
LD   A, R    ; se pregătește comutarea
XOR  80H     ;        înapoi, pentru întoarcerea
LD   R, A    ;        din subrutină
JP   (HL)    ; întoarcere în bancul apelant
```

În cazul concret al sistemului proiectat, deoarece am paginat bancul **RAM0** pe **EPROM** (în zona de adrese a primilor 32Ko), acest tip de apel este util mai ales în situația când **RAM0** nu se utilizează pentru a face "shadow" la memoria **EPROM** prin copiere, deci conținutul celor două memorii este diferit și se dorește, de exemplu, apelul unor rutine **BIOS** din **EPROM**, aflate la adresele fixate amintite. Memoria **EPROM** fiind "read only", stiva va trebui alocată în bancul **RAM1**, care nu este expandat, revenirea din subrutine putându-se realiza prin:

```
LD   A, R    ; se pregătește comutarea
XOR  80H     ;        înapoi, pentru întoarcerea
LD   R, A    ;        din subrutină
RET          ; întoarcere cu comutare și extragere din stiva comună
```

Din același motiv, dacă programul se rulează în zona superioară (adresele între 32Ko și 64Ko), paginarea și apelul rutinelor se poate face în mod obișnuit, fără nici una din metodele și precauțiile amintite (pierzându-se și avantajul paginării prin **R** față de paginarea clasică printr-un bit de port). Existența bancului comun celor două configurații aduce totuși avantajul posibilității de transfer între **EPROM** și **RAM0**, prin intermediul **RAM1**, asigurând astfel o flexibilitate mai mare sistemului.

Menționez ca ideea de utilizare a bitului 8 din registrul **R** am preluat-o din microcalculatorul "**COBRA**" **[13.]**, dar acolo era utilizată la schimbarea completă și într-un singur sens a configurației memoriei (chiar cu anumite erori hardware!), pe când în sistemul prezentat este utilizată la realizarea memoriei ***RAM – "shadow"*** pentru **EPROM**, cu posibilitate de salt și apel direct între cele două, în ambele sensuri. Am mai folosit această metodă de extindere a spațiului de adrese pentru microprocesorul **Z80** la dublarea memoriei **RAM** (de la

64Ko la 128Ko) la un microcalculator "**HC-91**", unde comutarea memoriei era completă, pe tot spațiul de adresare, posibilitatea de salt între bancuri fiind esențială.

În **figura 2.1.4.** se poate urmări conectarea circuitelor de memorie în sistem, iar **figura 2.1.5.** prezintă harta memoriei în spațiul de adrese al microprocesorului:

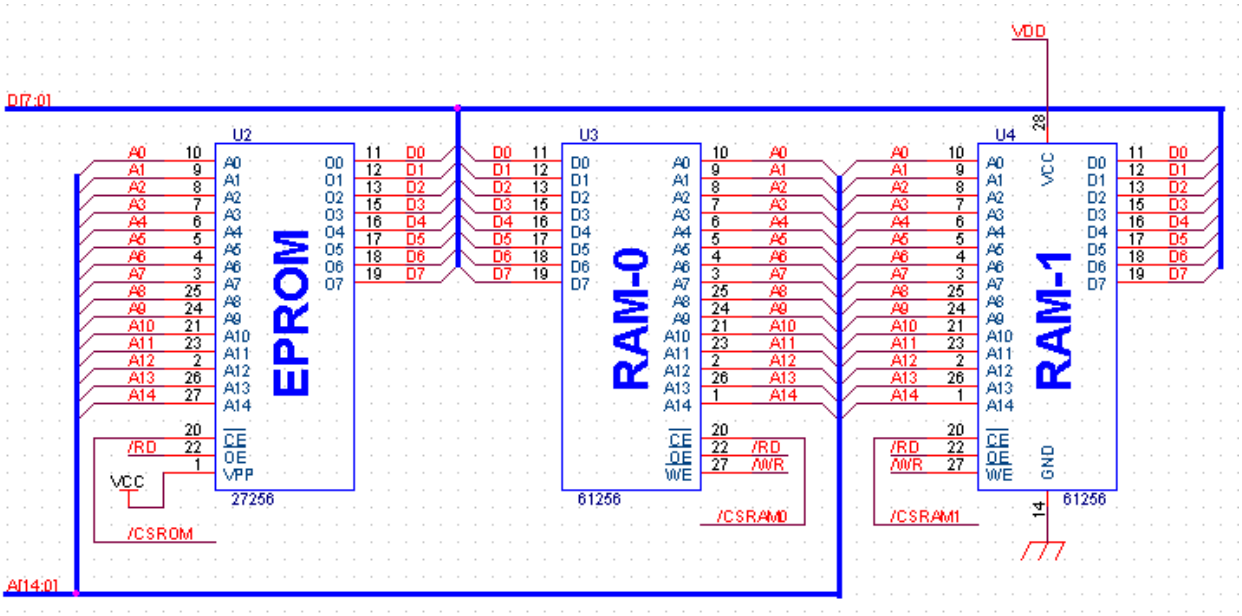


Fig. 2.1.4.

Memoria **EPROM** este un circuit **27C256** cu capacitatea de 32Ko × 8bit (realizat în tehnologie **CHMOS** rapidă, cu consum redus), ocupând prima jumătate a spațiului de adrese-memorie, iar pentru bancurile **RAM0** și **RAM1** am utilizat memorii **RAM** statice **61C256** cu aceeași capacitate (de tipul memoriilor "cache" pe plăcile cu **80486**). Utilizarea unor memorii dinamice mai vechi nu ar fi adus nici un avantaj sistemului, acestea complicând inutil schema de accesare și fiind extrem de sensibile la perturbațiile din medii industriale.

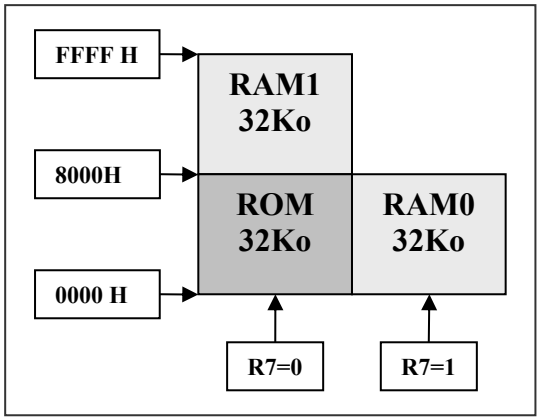


Fig. 2.1.5.

Magistralele tuturor circuitelor de memorie sunt conectate direct la magistralele procesorului, având un "fan-in" extrem de redus, datorită tehnologiei moderne. Validarea ieșirilor pe magistrala de date se realizează cu **/RD**, iar validarea scrierii în **RAM**-uri, cu **/WR**. Linia A15 care împarte spațiul de adrese al microprocesorului în două jumătăți se conectează separat, la blocul de decodificare pentru selecția memoriilor. La memoria **EPROM**, pinul pe

care se aplică tensiunea de programare (Vpp) este conectat la Vcc, iar memora **RAM1** are alimentarea separată, pe linia Vdd care provine din blocul de comutare pe baterie, pentru posibilitatea de păstrare a informației după deconectarea tensiunii Vcc.

Figura 2.1.6. prezintă modul de realizare a selecției memoriilor, inclusiv pentru paginarea **EPROM**-ului, după o schemă originală, în care rolul pricipal este deținut de cele două secțiuni ale decodorului **MSI** (cu integrare pe scară medie) – **HCT 139**, tot de tip **CHMOS** rapid **[15.]**. Din prima secțiune se obține semnalul de acces la memorie la ieșirea ***Y2A*** (numai atunci când **/REFSH** este "**1**" și **/MREQ** este "**0**", pentru evitarea selectării duble a memoriilor în ciclul "fetch" care ar fi fost deranjantă pentru timpii de acces, chiar dacă la reâmprospătare nu se generează **/RD** sau **/WR**), precum și semnalul de strob negativ al adresei de "refresh" la ieșirea ***Y0A*** (când **/REFSH** este deja "**0**" și **/MREQ** trece și el pe "**0**").

Acest mod de predecodificare elegant conduce la reducerea hazardului atât la selecția pentru "fetch", cât și la strobarea bitului 7 al registrului **R** pentru expandare. Tocmai aici, schema logică a microcalculatorului "**COBRA**" lăsa de dorit, strob-ul fiind obținut prin întârzierea frontului negativ pe **/REFSH**, fără condiționarea suplimentară cu **/MREQ**, ceea ce ducea uneori la porniri eronate (adresa de refresh A7 nu era perfect stabilă). Bitul 7 din registrul **R** este memorat într-un bistabil "D" – **LS 74** pe frontul pozitiv obținut prin inversarea strob-ului negativ de la ieșirea ***Y0A***.

Cea de-a doua secțiune **HCT 139** decodifică toate combinațiile dintre linia de adresă A15 și ieșirea Q a bistabilului, validate prin semnalul de acces la memorie (ieșirea **Y2A**), după cum se poate vedea și din tabelul următor:

/E (= Y2A)	**B (= Q)**	**A (= A15)**	**IEȘIRE ACTIVATĂ:**
1	x	x	– nici una (nu este accesată memoria)
0	0	0	Y0B = **/CSROM**
0	0	1	Y1B = **/CSRAM1_0**
0	1	0	Y2B = **/CSRAM0**
0	1	1	Y3B = **/CSRAM1_1**

Se observă că datorită decodificării complete, se obțin două semnale de selecție pentru **RAM1** (**/CSRAM1_0** și **/CSRAM1_1**), care trebuie unificate ulterior printr-un nivel de SAU într-unul singur (**/CSRAM1**), deoarece bancul **RAM1** este comun ambelor configurații de paginare. Pentru a fi posibilă o eventuală extindere în exterior a memoriei sistemului (prin paginare clasică pe porturi), am prevăzut un semnal de invalidare a bancului **RAM1** din exterior, provenit de la conectorul de extensie. Cele două porți **NAND** din schemă realizează atât

unificarea semnalelor de la decodor (active *LOW*) pentru selecția bancului, cât și invalidarea acestei selecții când semnalul exterior **/RAMD** = ”RAM1 Disable” este forțat pe “0” logic. Deoarece timpul de propagare prin cele două secțiuni cascadate ale decodorului este destul de mare, aici am montat un circuit **ALS 00**, care comută mai rapid decât porțile **LS** sau **HCT**, pentru a nu mai introduce întârzieri suplimentare **[9.]**.

La resetarea sistemului, bistabilul este forțat pe “0” prin intrarea asincronă de ștergere, sistemul pornind astfel întotdeauna pe **EPROM**. Imediat după primul “fetch”, bistabilul va fi încărcat tot cu valoarea “0”, deoarece registrul **R** este șters și el la resetarea microprocesorului.

Menționez că toate porțile și bistabilele rămase disponibile pe capsule le-am folosit complet în cadrul altor blocuri funcționale ale sistemului, unde erau absolut necesare (nu există utilizări “de umplutură”).

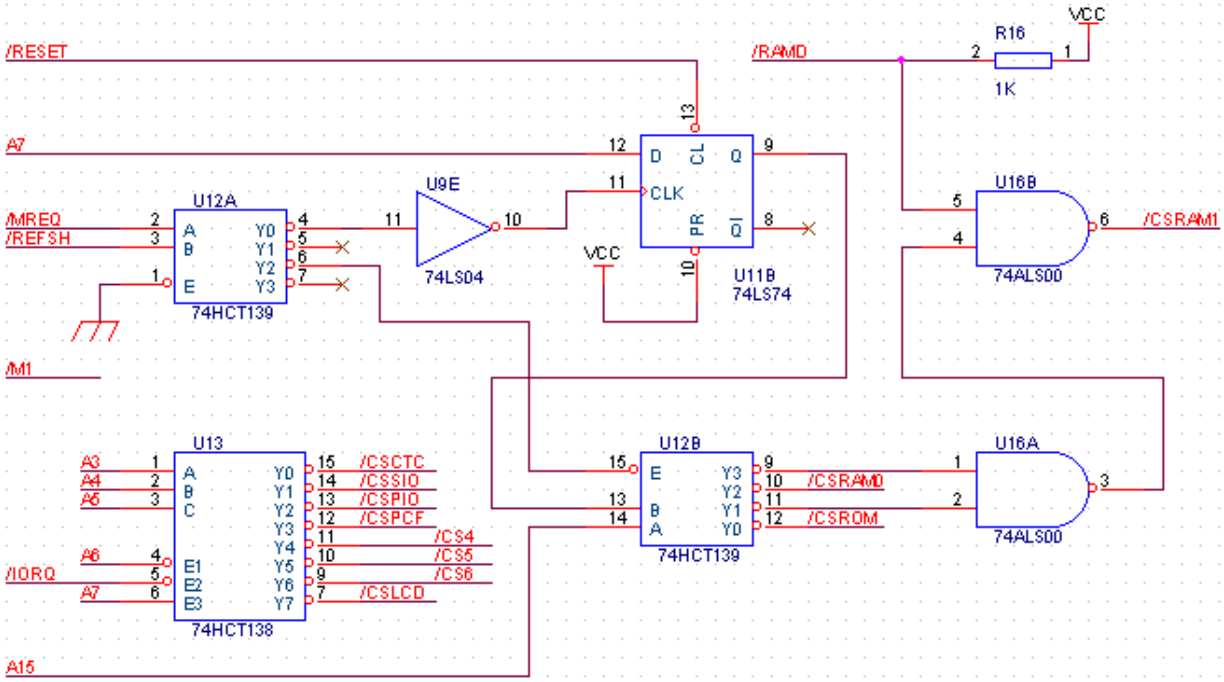


Fig. 2.1.6.

Într-un sistem cu microprocesor **Z80**, lucrul cu dispozitivele periferice se face printr-un ***ciclu-mașină de intrare/ieșire***, marcat de activarea semnalului **/IORQ** care se folosește împreună cu linii de adresă la selecția dispozitivelor **I/O**, însoțit de **/RD** sau **/WR** care specifică direcția transferului pe magistrală (**fig. 2.1.7.**). Microprocesorul acordă un timp suplimentar față de ciclii “fetch” și de acces la memorie pentru stabilizarea adresei, activând semnalul **/IORQ** la o perioadă de tact după depunerea adreselor pe magistrală. Pentru a permite dispozitivului periferic

să se selecteze în timp util, **Z80** inserează în mod automat și o perioadă de tact de așteptare (Tw), care prelungește ciclul de acces.

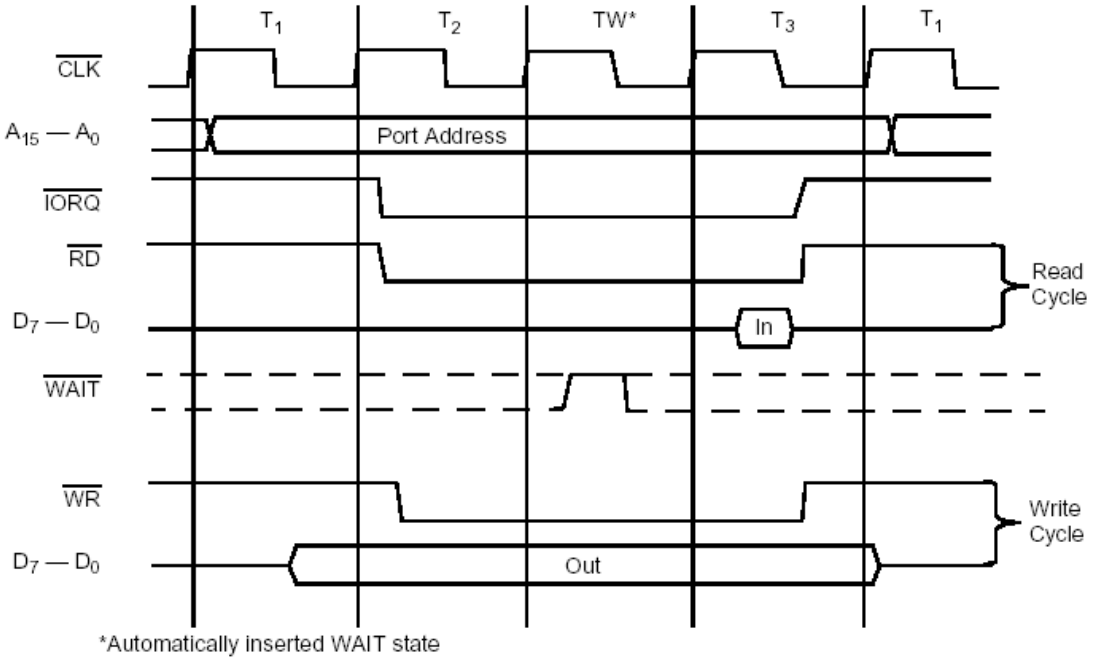


Fig. 2.1.7.

Am prezentat și această diagramă de timp **[16.]** deoarece este necesară pentru explicarea modului de conectare a controllerului **I2C** în sistem (vezi **&2.6.**).

Este interesant de subliniat că prezența celor două semnale distincte **/MREQ** și **/IORQ** în locul unuia singur ($M/\overline{IO}$ la **I8086 [10.]**) permite selectarea comodă și elegantă a circuitelor din spațiul de adrese-memorie și a celor din spațiul de adrese-intrare/ieșire, prin validarea separată a acestora, numai în momentul transferului.

În ***<u>ciclul de întrerupere mascabilă</u>*** procesorul acceptă întreruperea semnalată pe linia **/INT** printr-un "fetch" de preluare a vectorului de întrerupere, activând **/M1** și **/IORQ** în locul lui **/MREQ** de la ciclul "fetch" propriu-zis. De această dată însă, nu se activează semnalul **/RD**, citirea vectorului fiind strobată prin **/IORQ**, care apare cu 2.5 stări T decalat față de **/M1**, inserându-se automat 2 stări de *WAIT*, Tw. Se acordă astfel timp suplimentar pentru propagarea și stabilizarea priorităților prin <u>*maxim 4 circuite periferice*</u> conectate în lanțul de priorități serie specific familiei **Z80**. Ca și în "fetch"-ul propriu-zis, ciclul se termină cu 2 tacți de reâmprospătare a memoriilor dinamice, cu activarea **/REFSH** și **/MREQ [2.]**.

Problema care se pune la selecția circuitelor periferire este asigurarea compatibilității între circuitele din familia **Z80** care lucrează cu magistrala într-un mod mai particular și

circuitele din alte familii, deoarece circuitul de interfață **I2C** și magistrala pentru **LCD** lucrează în modul de magistrală Intel **82xx** (mai universală). De asemenea, sistemul este în așa fel gândit și proiectat, încât să permită extinderea în exterior cu circuite periferice din familia **I82xx [6.]**.

Majoritatea circuitelor de tip periferic-integrat au cel puțin una din intrările **/RD** sau **/WR**, dar nu toate au și intrarea **/IORQ**. La circuitele din familia **Z80** apare numai **/RD**, acestea generând **/WR**-ul intern, iar intrarea **/IORQ** este prezentă, fiind folosită și la recunoașterea ciclului de achitare-întrerupere (când apare însoțit de activarea lui **/M1**, dar fără **/RD** sau **/WR**). De asemenea, aceste circuite își reiniționalizează logica serială de priorități prin detectarea extragerii instrucțiunii ***RETI*** de către procesor, folosind apariția lui **/M1** însoțit de **/RD**, cu **/IORQ** inactiv.

Circuitele din familia **82xx**, au ambele intrări **/RD** și **/WR**, dar nu au **/IORQ**. Deoarece aceste circuite nu plasează pe magistrala de date un vector de întrerupere, s-ar părea că la selecția cipului ar trebui decodificată și absența activării lui **/M1**, pentru a evita o eventuală selectare parazită. Totuși, acest lucru nu este necesar, deoarece în lipsa unuia din semnalele **/RD** și **/WR**, circuitele ***Intel***, își păstrează starea de impedanță ridicată în ciuda unui eventual semnal de selecție favorabil, iar în ciclul de achitare-întrerupere **/RD** sau **/WR** nu se activează. De fapt, la microprocesoarele Intel (**I8080 [4.]**, **I8086 [3.]**) semnalele **/RD** și **/WR** ale circuitelor periferice se leagă la liniile **/IORD**, respectiv **/IOWR**, care se activează numai în cazul ciclilor de intrare-ieșire. O primă posibilitate de adaptare la magistrala lui **Z80** a acestor circuite ar fi refacerea acestor semnale, prin condiționarea **/RD** și **/WR** cu **/IORQ** (folosind câte o poartă "OR", semnalele fiind active pe "0" logic): $\overline{IOR} = \overline{RD} + \overline{IORQ}$;

$$\overline{IOW} = \overline{WR} + \overline{IORQ}.$$

Totuși, această metodă necesită porți suplimentare și nu permite utilizarea unui singur decodificator pentru toate perifericele din sistem, motiv pentru care am apelat la a doua variantă: condiționarea cu **/IORQ** a semnalelor **/CS** și legarea directă a intrărilor **/RD** și **/WR** la liniile corespunzătoare ale procesorului. Metoda nu compromite buna funcționare a circuitelor periferice din familia **Z80**, prezența lui **/IORQ** ca intrare directă în aceste circuite fiind de fapt o condiționare redundantă. La operațiile normale de **I/O** semnalul era oricum activ, iar în ciclul de recunoaștere a întreruperii și la reiniționalizarea logicii de priorități, circuitul nu trebuie să fie selectat. Folosind această condiționare, am reușit să utilizez un singur decodificator cu validare de tip **HCT 138 [15.]**, pentru generarea tuturor "chip select"-urilor pentru circuitele de **I/O** din sistem și chiar a 3 **/CS**-uri adiționale, disponibile la conectorul de extensie (**fig. 2.1.6.**).

Decodificatorul este validat de combinația de semnale **(A6=0, /IORQ=0, A7=1)** pe intrările de validare **(/E1, /E2, E3)**, pentru decodificare folosindu-se biții de adresă **(A5A4A3)**

conectați în această ordine la intrările **(CBA)**. În acest fel, am reușit decodificarea completă a 5 biți de adresă, ultimii 3 biți mai puțin semnificativi rămânând disponibili pentru selectarea a 8 porturi posibile dintr-un circuit periferic, valoare mai mult decât suficientă pentru conectarea unor periferice complexe, cum ar fi chiar un "hard-disk" cu controller **IDE** (care are nevoie de 3 biți de adresă și două semnale de "chip select").

Din acest motiv, circuitele care au numai 2 sau 4 porturi vor ocupa de fapt tot 8 porturi, prin repetarea porturilor în ordine, pe întreaga zonă de adrese definită prin **/CS**-ul respectiv (datorită faptului că astfel de circuite prezintă mai puțini biți de selecție internă, din cei 3 biți disponibili rămânând câțiva indiferenți). Maparea adreselor de port în spațiul de **I/O** considerat numai pe 8 biți (microprocesorul **Z80** permite și adresarea dispozitivelor de intrare-ieșire pe 16 biți), dedusă din modul de alocare al ieșirilor decodificatorului pe semnalele de selecție, este cea din tabelul următor:

ADRESE DE PORT I/O MAPATE:	**UTILIZARE:**
80h, 00, **81h,** 01, **82h,** 02, **83h,** 03,	Temporizatoare/numărătoare – **Z80 CTC**
84h, 04, **85h,** 05, **86h,** 06, **87h,** 07,	Circuitul **Z80 CTC** repetat (cu: A2=1)
88h, 08, **89h,** 09, **8Ah,** 0A, **8Bh,** 0B,	Interfața serială **RS 232** – **Z80 SIO**
8Ch, 0C, **8Dh,** 0D, **8Eh,** 0E, **8Fh,** 0F,	Circuitul **Z80 SIO** repetat (cu: A2=1)
90h, 10, **91h,** 11, **92h,** 12, **93h,** 13,	Interfața paralelă – **Z80 PIO**
94h, 14, **95h,** 15, **96h,** 16, **97h,** 17,	Circuitul **Z80 PIO** repetat (cu: A2=1)
98h, 18, **99h,** 19,	Interfața serială **I2C** – **PCF 8584**
9Ah, 1A, **9Bh,** 1B,	Circuitul **PCF 8584** repetat (cu: A2A1=01)
9Ch, 1C, **9Dh,** 1D,	Circuitul **PCF 8484** repetat (cu: A2A1=10)
9Eh, 1E, **9Fh,** 1F,	Circuitul **PCF 8584** repetat (cu: A2A1=11)
A0h, A1h, A2h, A3h, A4h, A5h, A6h, A7h,	Disponibile pentru periferic extern – **/CS4**
20, 21, 22, 23, 24, 25, 26, 27,	
A8h, A9h, AAh, ABh, ACh, ADh, AEh, AFh,	Disponibile pentru periferic extern – **/CS5**
28, 29, 2A, 2B, 2C, 2D, 2E, 2F,	
B0h, B1h, B2h, B3h, B4h, B5h, B6h, B7h,	Disponibile pentru periferic extern – **/CS6**
30, 31, 32, 33, 34, 35, 36, 37,	
B8h, 38, **B9h,** 39,	Interfața pentru afișaj alfanumeric – **LCD**
BAh, 3A, **BBh,** 3B,	Controllerul **LCD** repetat (cu: A2A1=01)
BCh, 3C, **BDh,** 3D,	Controllerul **LCD** repetat (cu: A2A1=10)
BEh, 3E, **BFh,** 3F,	Controllerul **LCD** repetat (cu: A2A1=11)

C0h ÷ FFh = FREE
40h ÷ 7Fh = FREE
⟹ 128 I/O PORTS

(* = FOR LCD)

2.2. Blocul generator de tact și obținerea frecvențelor pentru interfața serială.

Generatorul de tact are rolul de a furniza semnalul de ceas (**CLK**) către microprocesor și circuitele periferice care necesită prezența acestui semnal pentru secvențierea unor operații interne și pentru sincronizarea transferurilor pe magistrale. De asemenea, blocul produce și un semnal primar (**BAUD**), care, prin divizări suplimentare programabile, permite formarea ulterioară a frecvențelor de transmisie și recepție pe interfața serială **RS-232**.

Microprocesorul **Z80B** și circuitele sale periferice funcționează la o frecvență maximă de tact cu valoarea de **6.17 MHz [17.]**, fapt care permite utilizarea convenabilă a unei frecvențe pentru ceasul sistemului de **6.144 MHz**, obținută prin divizarea cu **2** a semnalului de la un oscilator cu quartz pe **12.288 MHz**. Această frecvență, descompusă în factori primi, conține majoritatea termenilor necesari pentru formarea ratelor uzuale de transmisie pe interfața serială **[3.]**:

$$6.144 MHz = 6144000 Hz = 2^{14} \bullet 3 \bullet 5^3 Hz .$$

Rata serială (BAUDS, BPS)	Descompunerea în factori primi	Divizarea necesară (în factori)	Divizarea calculată
300	$= 2^2 \bullet 3 \bullet 5^2$	$\div 2^{12} \bullet 5$	20480
600	$= 2^3 \bullet 3 \bullet 5^2$	$\div 2^{11} \bullet 5$	10240
1200	$= 2^4 \bullet 3 \bullet 5^2$	$\div 2^{10} \bullet 5$	5120
2400	$= 2^5 \bullet 3 \bullet 5^2$	$\div 2^9 \bullet 5$	2560
4800	$= 2^6 \bullet 3 \bullet 5^2$	$\div 2^8 \bullet 5$	1280
9600	$= 2^7 \bullet 3 \bullet 5^2$	$\div 2^7 \bullet 5$	640
14400	$= 2^6 \bullet 3^2 \bullet 5^2$	nu se poate obține	–
19200	$= 2^8 \bullet 3 \bullet 5^2$	$\div 2^6 \bullet 5$	320
28800	$= 2^7 \bullet 3^2 \bullet 5^2$	nu se poate obține	–
38400	$= 2^9 \bullet 3 \bullet 5^2$	$\div 2^5 \bullet 5$	160
57600	$= 2^8 \bullet 3^2 \bullet 5^2$	nu se poate obține	–
115200	$= 2^9 \bullet 3^2 \bullet 5^3$	nu se poate obține	–

Din tabelul prezentat anterior, se observă că ratele de transfer: 14400, 28800, 57600, 115200 bauds, nu se pot obține din acest quartz, deoarece ar necesita prezența unui termen 3 suplimentar în descompunere.

Pentru obținerea divizărilor programabile am utilizat câte un canal numărător din circuitul periferic **Z80B-CTC**, pentru fiecare din cele două canale seriale. Aceste canale pe 8 biți pot

funcționa în mod temporizator, cu predivizor (1/16 sau 1/256 din ceasul sistemului), sau în mod numărător, fără predivizor. Deoarece, pentru sincronizarea de bit în modul de transmisie asincron, interfețele seriale necesită o frecvență de 16 ori mai mare decât rata de transmisie a datelor, cea mai mare frecvență disponibilă pentru serială este:

$$\frac{6144000}{16 \bullet 16} = \frac{2^{14} \bullet 3 \bullet 5^3}{2^4 \bullet 2^4} = 2^6 \bullet 3 \bullet 5^2,$$

de unde rezultă că cea mai mare rată serială ce se poate obține folosind canalul în modul temporizator este de:

$$\frac{2^6 \bullet 3 \bullet 5^2}{5} = 2^6 \bullet 3 \bullet 5 = 4800 baud,$$

rată ce se obține prin încărcarea canalului cu constanta 5.

Această rată este prea mică, din considerente de versatilitate a sistemului, deci utilizarea canalelor in modul temporizator cu predivizare nu este avantajoasă. Dacă acestea se folosesc în modul numărător, impulsurile primite pe intrările de numărare trebuie să aibă perioada de cel puțin 2 ori mai mare decât a ceasului sistemului, deci este necesar un predivizor extern. Predivizorul, împreună cu canalul numărător programabil, trebuie sa elimine tremeni de forma $2^n \bullet 5$, astfel încât să fie posibilă obținerea a cât mai multe rate de baud uzuale.

Pentru obținerea unei rate de transfer cât mai mari, predivizorul extern trebuie să elimine numai constanta 5 din descompunerea în factori a ceasului de **6.144 MHz**, divizarea suplimentară prin puteri ale lui 2 rămânând în sarcina canalului, la opțiunea programatorului. Deoarece un termen 5 trebuie oricum eliminat, indiferent de rata de baud (după cum se observă din tabelul anterior), iar eliminarea oricărui termen – putere a lui 2 din descompunere reduce rata maximă, utilizarea unui predivizor extern cu 5 reprezintă soluția optimă.

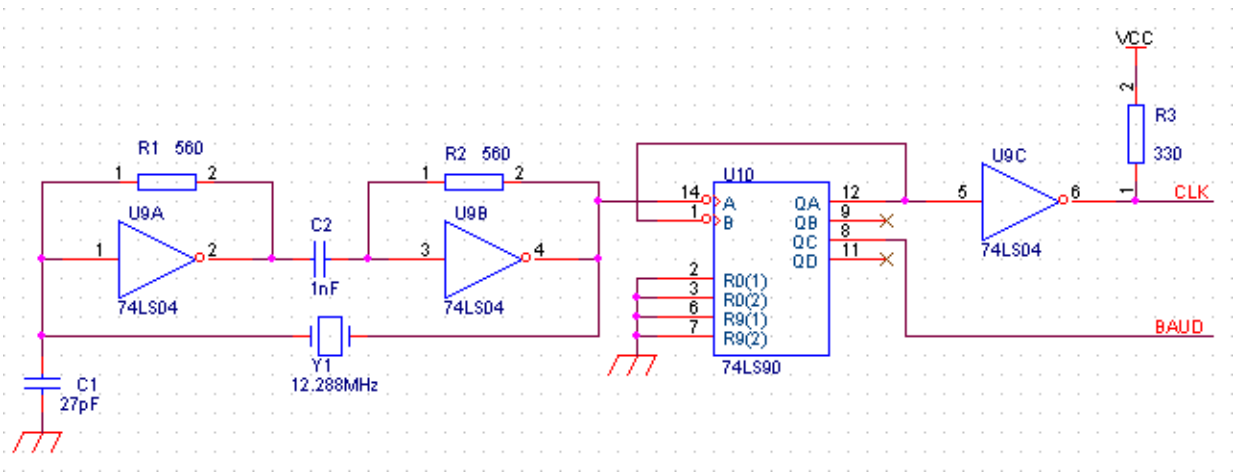


Fig. 2.2.1.

În **figura 2.2.1.** este prezentată schema blocului generator, care produce semnalele **CLK** și **BAUD**. Oscilatorul este de tip serie, fiind realizat cu inversoare **LS-TTL** care lucrează în regim de amplificator (prin polarizarea de tip reacție negativă în curent continuu cu rezistențele R1, respectiv R2, în zona activă a caracteristicii de tranfer), după o schemă clasică **[9.]**. Rezistoarele nu sunt critice, valoarea de 560Ω fiind optimă pentru inversoarele din seria **LS**, iar condensatorul C2, de cuplaj între amplificatoare, are o valoare suficient de mare pentru a se comporta ca un scurtcircuit la frecvența de lucru.

Montajul de tip serie are câteva avantaje. Quartz-ul lucrează exact pe frecvența de rezonanță serie, care corespunde frecvenței de rezonanță mecanică a cristalului (marcată în clar), independentă de capacitățile parazite externe cristalului, spre deosebire de frecvența paralelă ce depinde de rezonanța electrică a circuitului echivalent. Alt avantaj al schemei cu rezonator serie este stabilitatea mai bună datorată unui factor de reacție pozitivă mai puternic (2 inversoare) și a defazajului mai precis (180°), realizat numai cu componentele active, fără condensatoare suplimenare, ca în cazul oscilatorului de tip paralel. De asemenea, factorul de transmisie pe bucla de reacție, mult supraunitar, permite o ușoară "dreptunghiularizare" a semnalului încă de la ieșirea oscilatorului, care nu mai lucrează sinusoidal.

Singura observație ce trebuie amintită este prezența condensatorului "de pornire" – C1. Am observat, în urma unor teste, că lipsa acestui condensator nu asigură o amorsare sigură a oscilatorului la conectarea tensiunii de alimentare. Explicația rezidă în simetria schemei, care poate ajunge la un echilibru în curent continuu în lipsa unui semnal de excitație inițial (eventual cu mici oscilații în jurul unei valori de curent continuu aflată între nivelele de "0" și "1" logic, cu perioada dată de timpul de propagare, atunci când caracteristica inversoarelor este foarte abruptă sau prezintă histerezis). Condensatorul C1 asigură amorsarea prin pornirea de la tensiune zero a intrării oscilatorului, care nu mai poate ajunge astfel la echilibru în curent continuu. Valoarea sa este suficient de mică, astfel încât încărcarea și descărcarea sa sunt mult mai scurte decât semiperioada semnalului dreptunghiular generat de oscilator.

Din motivele prezentate, pentru generarea semnalului primar pentru interfața serială, am utilizat un circuit **LS 90** care conține patru bistabile JK "master-slave", interconectate astfel încât să realizeze un divizor cu 2 și un divizor cu 5, cu intrări comune de resetare (neutilizate în schemă) **[14.]**. Numărătorul - divizor cu 2 primește semnal direct de la oscilator, intrarea sa de ceas neavând o încărcare prea mare (1,5 sarcini LS-TTL), la ieșirea lui obținându-se frecvența de ceas a sistemului, $\mathbf{f_{CLK}=6.144\ MHz}$. Acest semnal este apoi trecut printr-un inversor suplimentar, pentru mărirea "fan-out"-ului, prevăzut la ieșire și cu un rezistor de tip "pull-up" la Vcc, de valoare relativ mică, pentru formarea nivelului de "1" logic necesar circuitelor **MOS** din familia

microprocesorului **Z80 [2.]** (acest rezistor are și rolul de îmbunătățire a frontului pozitiv, care este mai puțin abrupt decât cel negativ în cazul circuitelor de tip **TTL**).

Numărătorul modulo 5 preia semnalul direct de la ieșirea divizorului cu 2, pentru a nu încărca suplimentar linia de tact (inversarea față de semnalul **CLK** neavând importanță) și furnizează la ieșirile Qc și Qd semnale cu perioada de 5 ori mai mare, dar cu factori de umplere diferiți: 2/5, respectiv 1/5. Pentru generarea frecvențelor de **BAUD**, am preferat semnalul cu un factor de umplere mai mare, deoarece acesta trebuie aplicat în continuare circuitului numărător programabil și este mai puțin critic în privința constrângerilor de timp decât cel cu factorul de umplere 1/5.

2.3. Blocul numărător – temporizator programabil și controler de întreruperi.

Acest bloc are rolul de a asigura sistemului o referință de timp real (întrerupere la un interval programabil bine determinat), posibilitatea măsurării precise (prin hardware) a unor intervale de timp, posibilități de numărare a unor evenimente externe (utile mai ales în cazul unor aplicații de control automat), generarea ratelor programabile de transmisie - recepție pentru interfața serială, precum și adaptarea / arbitrarea cererilor de întrerupere de la circuite care nu fac parte din familia **Z80 [6.]** (de exemplu controllerul **I2C – PCF 8584**, sau alte circuite externe sistemului). Pentru aceste circuite, se asigură alocarea poziției (fixe) în lanțul de priorități și generarea vectorului de întrerupere în modul vectorizat avansat, specific microprocesorului **Z80** (**modul 2**).

Blocul este, de fapt, un circuit temporizator - numărător **Z80B-CTC**, conectat în schemă astfel încât să poată fi utilizat într-un mod cât mai universal posibil.

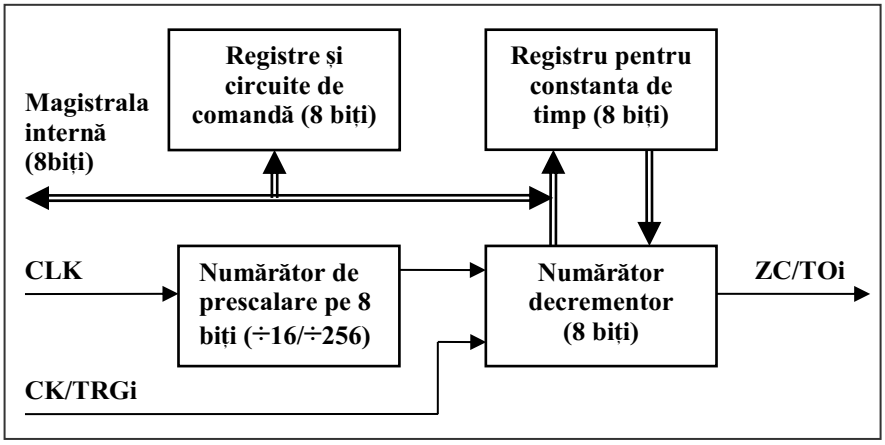


Fig. 2.3.1.

Pe scurt, acest circuit **[2.]** are patru canale de tip "timer/counter" care funcționează prin decrementare pe 8 biți, cu posibilitate de prescalare în modul temporizator (predivizare cu 16 sau cu 256 a ceasului sistemului) și cu reâncărcare automată în numărătorul - decrementor a constantei de timp (înscrisă și păstrată într-un registru auxliar), la inițializare și la fiecare trecere prin zero. Structura bloc pentru un canal este cea din **figura 2.3.1.**

În modul temporizator se folosește ceasul sistemului (**CLK**), cu prescalare și cu declanșare automată sau externă (prin intrarea **TRG**), iar în modul numărător, canalul numără fronturile active de la intrarea externă de numărare (**CK**). Valoarea rămasă în numărător poate fi citită în orice moment de către unitatea centrală, fără a perturba funcționarea canalului, iar la trecerea prin zero apare un impuls pozitiv pe ieșirea **ZC/TO** (atât în modul temporizator – **TO**, cât și în modul numărător – **ZC**) și se face o cerere de întrerupere către unitatea centrală, dacă întreruperile canalului au fost validate. Circuitul modifică automat, pentru fiecare canal, vectorul de întrerupere (înscris inițial numai în canalul zero) și asigură o arbitrare fixă a priorităților (în funcție de numărul de ordine al canalului, cu canalul 0 - cel mai prioritar).

Un avantaj important al circuitului, pe lângă structura sa mai simplă și modul mai comod de operare față de alte circuite cu funcții asemănătoare din alte familii (de exemplu circuitul **I8253 [6.]**), este posibilitatea programării frontului activ utilizat pentru declanșarea măsurării timpului sau la numărarea evenimentelor. De asemenea, **Z80-CTC** are 4 canale, față de numai 3 ale circuitului **I8253**, iar lățimea de 8 biți asigură o granularitate a timpului suficientă pentru aplicațiile curente. Dacă este necesară o granularitate mai fină, canalele se pot conecta în cascadă.

Circuitul poate fi folosit și drept controller de întreruperi pentru circuite care nu sunt compatibile cu modul de lucru vectorizat al familiei **Z80**, utilizând unul sau mai multe canale pentru conversia semnalului de întrerupere, celelalte canale putând păstra rolurile de măsurare a timpului sau de contoare de evenimente. Acest lucru este posibil prin programarea canalului respectiv în modul numărător, încărcat cu constanta de timp **01h**, datorită logicii interne de gestionare a întreruperilor precum și posibilității alegerii frontului activ al intrării de numărare (care devine în acest caz intrare de tip IRQ_n sau $\overline{IRQ_n}$). Canalul ales mai poate fi încă folosit ca temporizator cu declanșare internă, atunci când nu se folosește întreruperea de la perifericul conectat la acesta.

Dezavantajul unei astfel de structuri constă în faptul că prioritățile sunt fixe și nu pot fi modificate prin program, dar acesta este dezavantajul întregului sistem de întreruperi în modul specific microprocesorului **Z80**, unde prioritățile sunt fixate de poziția fizică a circuitelor în lanțul de întreruperi, prioritate cablată prin semnalele **IEI**, **IEO**.

Am folosit aceată posibilitate la proiectarea sistemului, pentru integrarea controllerului **I2C** în sistemul de întreruperi al microprocesorului **Z80**, folosind canalul 3, care nu are, oricum,

ieșire de numărare **ZC/TO** și este cel mai puțin prioritar. În lipsa acestui artificiu, schema logicii de gestionare a întreruperii de la controllerul **I2C** s-ar fi complicat semnificativ, chiar dacă circuitul **PCF 8584** poate genera un vector de întrerupere memorat intern **[18.]**. În cazul altor circuite (de exemplu cele din familia **I82xx**) care pot fi adăugate în exterior sistemului, folosirea circuitului **CTC** devine obligatorie.

Principalul dezavantaj al circuitului **Z80-CTC** este faptul că decrementarea numărătorului pentru fiecare canal are loc în mod sincron cu ceasul sistemului, **CLK** (circuitul **I8253** nu are acest dezavantaj). Din acest motiv, **Z80-CTC** nu poate fi folosit ca generator programabil de rate seriale folosind un semnal de la un oscilator cu quartz separat, diferit de cel al sistemului, fiind necesară folosirea uniu cristal particular pentru tactul microprocesorului, care să fie și intrare de numărare pentru divizorul programabil.

Conectarea circuitului în sistem este prezentată în **figura 2.3.2.** Se remarcă conectarea directă la magistrala microprocesorului, cu semnalele specifice, folosind pentru selectarea circuitului un semnal **/CSCTC**, generat la decodificatorul de mapare **I/O**, precum și selectarea canalelor cu biții cel mai puțin semnificativi de adresă. Circuitul este conectat în lanțul de întreruperi în poziția cea mai prioritară, semnalul **IEI** de validare a întrerupeilor sale fiind conectat printr-un rezistor la Vcc și scos în exterior, la conectorul de extensie al sistemului, în vederea unei eventuale conectări externe a unui circuit mai prioritar. Canalul 0 devine cel mai prioritar periferic din sistem, urmat de celelalte canale și abia apoi de circuitul de interfață serială, care va primi semnalul de validare a întreruperilor de la ieșirea **IEO**. Această mapare este necesară deoarece întreruperile de la referința de timp real sunt prioritare față de oricare altă sursă de întrerupere.

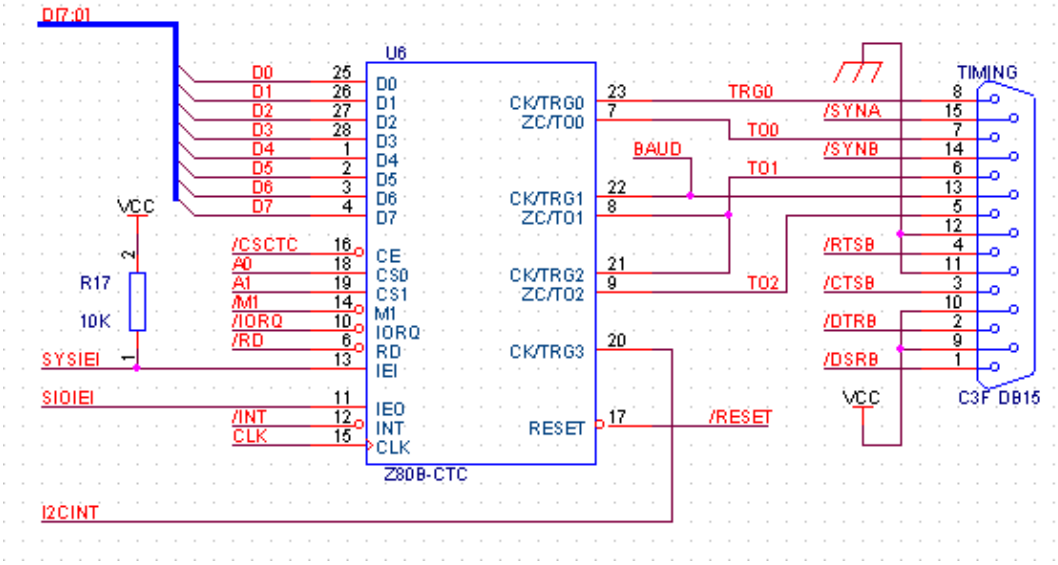


Fig. 2.3.2.

Semnalele canalului 0 sunt complet disponibile la conectorul cu 15 pini **C3F-DB15**, denumit "TIMING", canalele 1 și 2 sunt conectate în cascadă, fiind folosite pentru generarea ratelor pentru cele 2 canale de interfațare serială, iar canalul 3 redirectează întreruperea de la interfața **I2C**. Semnalele **TO1** și **TO2** sunt conectate și la intrările de ceas ale celor două canale de comunicație din interfața serială. S-ar părea că această conectare cablată în cascadă limitează posibilitățile de utilizare a circuitului, mai ales limitarea ratei maxime a canalului serial care primește semnalul de la cel de-al doilea canal **TO2**, care ar trebui să fie totdeauna mai mică sau cel mult egală cu viteza primului canal, ce are ceasul de shiftare de la **TO1**, dar aceste rate vor fi divizate suplimentar, eventual prin factori diferiți, în canalele seriale.

Semnalul **BAUD** aplicat la intrarea de numărare a canalului 1, prin felul în care a fost obținut (vezi **&2.2.**), îndeplinește atât constrângerea legată de mărimea perioadei (care trebuie să fie mai mare decât dublul perioadei de tact sistem), cât și constrângerea legată de lățimea impulsului (care trebuie să fie cel puțin o perioadă de tact sistem) **[17.]**.

Referința de timp real poate fi realizată folosind canalul 0, dar cascadarea cablată dă posibilitatea utilizării celor două canale (1 și 2) sau numai a ultimului ca o a doua referință de timp real, cu o frecvență de întrerupere foarte scăzută, dată de divizarea ceasului sistem sau a semnalului **BAUD** prin două canale succesive. Mai menționez că cel de-al doilea sau ambele canale pot fi încă utilizate independent, ca temporizatoare, în perioadele când nu se folosește interfața serială (în modul temporizator cu declanșare internă nu contează intrările **CK/TRG**). De asemenea, semnalele de sfârșit - numărare sunt scoase la conectorul **C3F-DB15**, împreună cu semnalul **BAUD**, pentru eventuale utilizări ca surse de impulsuri programabile, mai ales în situația când nu se folosesc ambele canale seriale.

2.4. Interfața serială - RS 232.

Pentru a putea controla sistemul de dezvoltare cu ajutorul unui calculator (pe care se pun la punct programele), pentru încărcarea unor programe în memoria **RAM** sau pentru comunicațiile cu alte sisteme în aplicații de tip control automat, este necesară prezența unei interfețe seriale. Cel mai răspândit standard pentru transfer serie a datelor este standardul asincron **RS 232**, care permite conectarea la porturile **COMx** ale unui calculator personal compatibil **IBM-PC**.

Cu ajutorul unui program monitor înscris în memoria **EPROM** a microsistemului și a unui program de tip terminal care rulează pe calculatorul **PC**, prin legătura serie se pot încărca și salva programele pentru sistem (în format **INTEL–HEX** sau direct în format binar), iar prin afișare în regim de tip consolă se poate vizualiza și modifica conținutul regiștrilor microprocesorului sau al memoriei, se pot citi sau înscrie porturile de intrare-ieșire, se pot depana programe prin "trasare" pas-cu-pas, iar dacă monitorul permite, se pot asambla și dezasambla programe în cod mnemonic direct pe microsistem ("on-the-fly").

În cazul aplicațiilor de tip control automat, prin legătura serială se poate comanda și controla de la distanță (de către un operator uman) funcționarea echipamentelor industriale conduse cu ajutorul nucleului de dezvoltare cu **Z80**, se pot prelua datele achiziționate de sistem de la senzori și traductoare, sau se pot conecta mai multe sisteme de conducere în rețea (realizate cu microprocesoare sau cu microcontrollere). În acest ultim caz, este necesară posibilitatea de conectare pe aceeași magistrală serială a mai multor echipamente care au facilități de recunoaștere automată a adresei (prin hardware).

Interfața serială a sistemului de dezvoltare este realizată în jurul unui circuit de interfață de tip **Z80-SIO/0** și are 2 canale de comunicație "full duplex", care pot fi folosite atât în mod asicron **RS 232**, dar și în mod sincron, cu posibilitate de conectare în rețea și recunoaștere automată a adresei. Circuitul este de fapt un controller serial "inteligent", foarte performant și cu posibilități multiple de operare **[2.]**. Cele două canale prezintă, pe lângă semnalele de transmisie/recepție serie și semnale pentru comanda și verificarea stării modemului, conform standardului **RS 232**, dar care pot fi folosite și ca ieșiri sau intrări de uz general, beneficiind de posibilitatea de întrerupere la schimbarea stării intrărilor ("interrupt-on-change", facilitate foarte utilă, asemănătoare porturilor microcontrollerului **PIC 16F84 [11.]**, care permite ca modificarea stării unui traductor să fie sesizată de sistem direct printr-o întrerupere și nu prin testare software a intrării - "pooling", nefiind necesar nici hardware suplimentar).

Pentru realizarea transferului de date, stări și comenzi cu unitatea centrală, **Z80-SIO** poate folosi testarea software ("pooling"), întreruperile, sau tehnica transferului pe blocuri de octeți. Circuitul apare în spațiul **I/O** prin patru porturi distincte, câte un port de date și unul de comenzi/stare pentru fiecare canal, resursele interne fiind accesibile prin adresare indirectă (se transmite mai întâi adresa registrului și apoi se transferă datele), pentru a nu încărca spațiul **I/O** cu adrese de registre interne și pinii cu semnale de selecție.

Tehnica testării software este utilă numai pentru programul monitor, presupunând verificarea permanentă sau periodică prin program a parametrilor ce caracterizează transferul, citind registrele de stare. Acest mod de lucru nu poate fi util în cazul unui sistem de operare în timp real, pentru control industrial, care este condus de evenimente și folosește întreruperi (este de tip "interrupt - driven").

Lucrul cu întreruperile este facilitat de prezența unui controller de întreruperi intern, care gestionează întreruperile din **8** surse, câte patru pentru fiecare canal ("buffer" de transmisie gol, schimbare stare intrare auxiliară, caracter recepționat disponibil, condiție specială de recepție), putând "puncta" direct opt rutine de tratare prin modificarea vectorului de întrerupere inițial, în funcție de sursă (lucrând în **modul 2**, vectorizat).

Interesantă este și posibilitatea transferului pe blocuri de octeți, folosind instrucțiunile cu repetare specifice procesorului ***Z80*** (***OTIR***, ***OTDR***, ***INIR***, ***INDR***) **[2.]**. Acest mod de operare necesită sincronizarea hardware a operațiilor de intrare/ieșire pentru a nu se pierde informațiile atunci când perifericul nu este pregătit sau nu are datele disponibile la viteza de lucru a microprocesorului. Pentru aceasta, cele două canale dispun de două linii care pot fi programate ca ieșiri de tip **/WAIT** cu drena în gol, pentru a putea fi conectate într-o configurație SAU-cablat (semnalele sunt active *LOW*) pe intrarea similară a microprocesorului.

De asemenea, modul de transfer pe blocuri se mai poate folosi și la programarea circuitului, prin completarea unei tabele cu adresa și conținutul fiecărui registru de comandă, care este apoi transmisă printr-o singură instrucțiune ***OTIR***, în acest caz nefiind necesară sincronizarea (resursele interne sunt disponibile la viteza de lucru a unității centrale).

Circuitul **Z80-SIO/0** are o singură intrare pentru ceasul de recepție / transmisie serială pe canalul B, având în schimb prezente toate semnalele pentru modem, spre deosebire de celelalte versiuni **Z80-SIO/1** și **Z80-SIO/2**, care au intrări de ceas separate, dar le lipsește câte unul din pinii de comandă/stare pe canalul B (din limitare la 40 pini pe capsulă) **[2.]**. Am considerat mai utilă prezența completă a acestor semnale, deoarece lucrul cu rate seriale diferite la transmisie și recepție este mai rar întâlnit în practică și era, oricum, imposibil datorită existenței a numai două surse de ceas programabil pentru **BAUD**, câte una pentru fiecare canal (ceasul de recepție este conectat împreună cu cel de transmisie și pentru canalul A, chiar dacă semnalele erau distincte).

Schema blocului de interfață serială se poate urmări în **figurile 2.4.1.** și **2.4.2.**

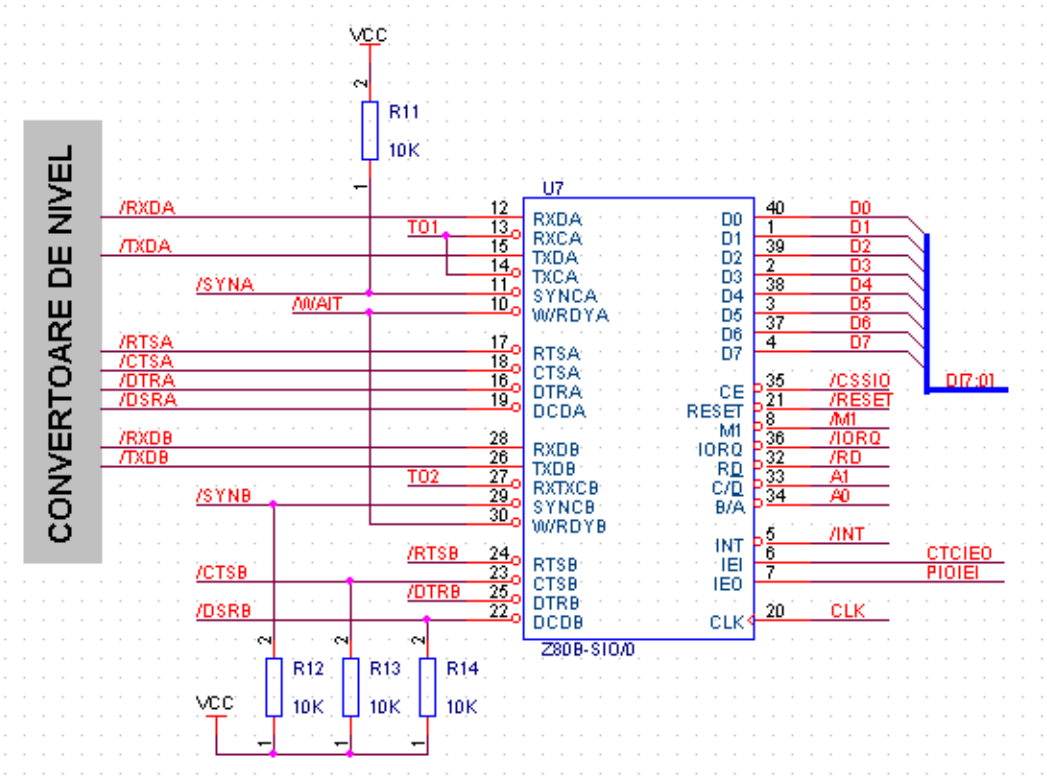


Fig. 2.4.1.

Se remarcă din **figura 2.4.1.** conectarea la magistralele specifice microprocesorului **Z80** în spațiul de **I/O**, folosind pentru selecția porturilor din circuit biții cel mai puțin semnificativi de adresă (**A0** – pentru selecție canal A=0/B=1 și **A1** – pentru selecție Date=0/Comenzi-stări=1) care, împreună cu semnalul de selectare a circuitului primit de la decodificatorul de adrese **I/O** (/**CSSIO**), asigură maparea porturilor la adrese succesive.

Controllerul serial este al doilea circuit în lanțul de priorități pentru întreruperi mascabile, primind semnalul de validare **IEI** de la **Z80-CTC**, și asigurând validarea întreruperilor pentru circuitul de interfață paralelă **Z80-PIO**, prin **IEO**. Această conectare relativ prioritară este necesară pentru evitarea pierderii caracterelor la recepție, situație care poate apărea mai ales în cazul ratelor mari de transfer folosind întreruperi, chiar dacă circuitul are câte un buffer **FIFO** de 4 nivele atât pentru date cât și pentru erori.

În modul asincron, **SIO** își formează perioadele de bit prin divizarea suplimentară cu 16, 32 sau 64 a semnalelor **TO1**, respectiv **TO2** primite de la **CTC**, divizare ce poate fi diferită pentru fiecare canal în parte. Acest fapt asigură obținerea ratelor mai mari pe canalul B față de canalul A, în anumite limite, chiar dacă ceasul **TO2** are totdeauna o frecvență mai mică decât

TO1 (vezi **&2.3.**). În modurile sicrone, rata de bit este egală cu frecvența semnalelor de ceas, acestea nefiind divizate suplimentar.

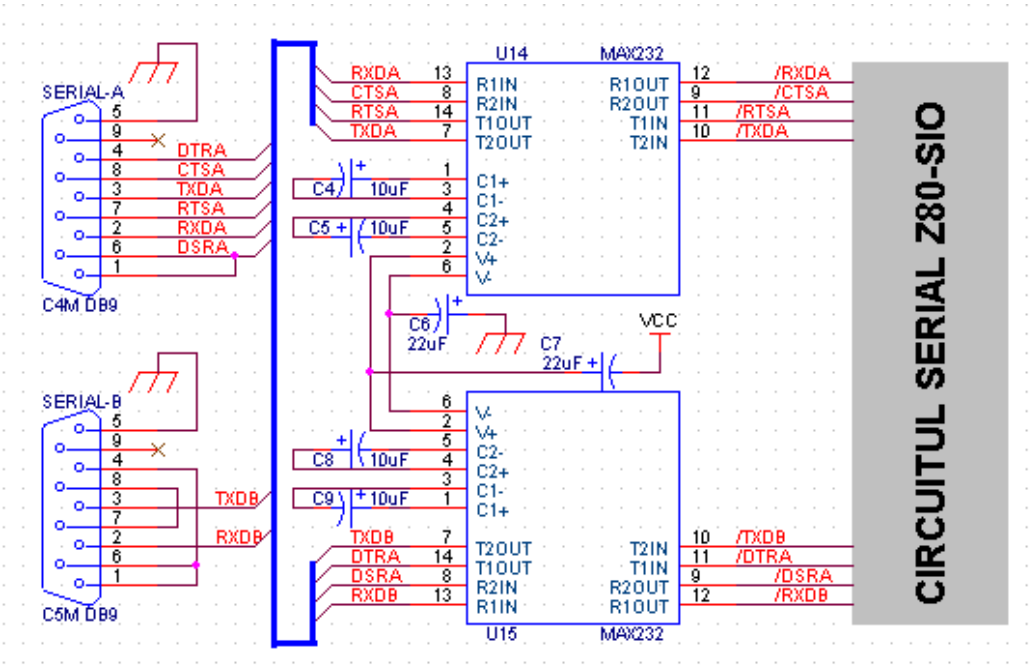


Fig. 2.4.2.

Pentru conversia nivelului de tensiune necesară standardului **RS 232** am utilizat (**fig. 2.4.2.**) două circuite **MAX 232** (**ICL 232 [21.]**), care permit alimentarea numai de la sursa Vcc=5V, generând intern tensiunile de alimentare a transmițătoarelor de linie (+10V și -10V) printr-o rederesare cu multiplicare de tensiune și inversare de polaritate a semnalului de la un oscilator **RC** intern (**fig. 2.4.3.**).

Condensatoarele C4 și C5, respectiv C8 și C9 din schemă sunt necesare dublorului de tensiune intern și inversorului de polaritate din fiecare circuit (condensatoare - pompă de sarcină), dar condensatoarele C6 și C7 (condensatoare - rezervor de sarcină, pe care se obțin tensiunile finale – negativă față de masă, respectiv pozitivă față de Vcc) le-am conectat în comun ambelor circuite, asigurând astfel pe lângă economia de componente pe cablaj și o filtrare mai bună a tensiunii redresate, cu scăderea impedanței sursei de alimentare, fapt care duce la creșterea capabilității în curent a inversoarelor de transmisie pe linie.

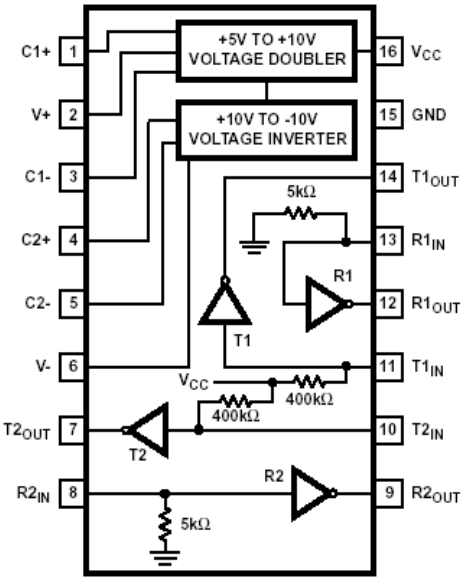


Fig. 2.4.3.

Alte caracteristici notabile ale circuitelor **s**unt protecția la scurtcircuit a inversoarelor de transmisie pe linie și plaja largă de tensiuni la intrarea în inversoarele de recepție (±30V), care prezintă și histerezis pentru creșterea imunității la zgomotele de pe liniile seriale **[21.]**.

Semnalele de comandă/stare ale canalului A sunt conectate în totalitate (prin intermediul convertoarelor de nivel – **fig. 2.4.2.**) la conectorul **C4M-DB9**, pentru a putea fi utilizate la conectarea unui modem sau la comunicația serială cu alt echipament de calcul folosind protocol hardware pentru controlul fluxului de date (**RTS/CTS** = ”Request To Send/Clear To Send” sau **DTR/DSR** = ”Data Terminal Ready/Data Set Ready”) **[6.]**.

Linia **DCD**A = ”Data Carrier Detect” (pinul nr. 1 de la conectorul **SERIAL-A**) se conectează de obicei împreună cu **DSR**A = ”Data Set Ready” (când se detectează purtătoarea modemul este automat pregătit pentru comunicație), iar linia **RI**A = ”Ring Indicator” a rămas nefolosită (în lipsa unui receptor de linie suplimentar prin intermediul căruia se putea conecta la intrarea/ieșirea de sincronizare **SYN**A, care poate fi utilizată drept intrare de uz general în modul asincron).

Canalul B are scoase la conectorul **SERIAL-B** numai liniile de transmisie/recepție serială **TXD**B și **RXD**B, pinii corespunzători semnalelor de protocol fiind conectați direct între ei, pentru situația când echipamentul necesită obligatoriu validarea transmisiei și recepției prin protocol hardware și această conectare nu este deja făcută în interiorul mufei corespunzătoare cablului de legătură (**RTS**B este conectat cu **CTS**B, iar **DSR**B împreună cu **DCD**B sunt legate la **DTR**B, astfel încât are loc auto-validarea transmisiei și recepției). În mod normal, pentru legătura cu alte echipamente de tip microsistem (cu microprocesor sau cu microcontroller) se folosesc numai 3 linii: **TXD**, **RXD** și **GND**, iar pentru controlul fluxului de date se utilizează un protocol soft – ***XON/XOFF***, beneficiind în mod esențial de existența canalului “full-duplex” **[6.]**.

Liniile sunt astfel împărțite între cele două circuite convertoare de nivel astfel încât funcționarea canalului A să necesite prezența obligatorie numai a unui singur circuit **MAX 232** (și anume **U14**), asigurând, pe lângă **TXD**A/**RXD**A, chiar și semnalele esențiale pentru controlul hardware al fluxului de date (**RTS/CTS**). Am realizat această împărțire pentru o eventuală scoatere din soclu a circuitului **U15** și înlocuirea lui cu un soclu “șrapat” ce permite trecerea semnalelor în mod direct, fără conversie de nivel, pentru utilizarea canalului B în modul sincron.

Liniile de control-modem ale canalului B împreună cu liniile de sincronizare pentru ambele canale sunt scoase la conectorul “TIMING” (**fig. 2.3.2.**), pentru utilizare generală, separat sau împreună cu semnalele blocului temporizator/numărător. Pentru a evita generarea unor întreruperi externe / de stare parazite atunci când una din linii rămâne nefolosită, am montat rezistențe de “pull-up” la Vcc pe toate aceste linii care sunt intrări. De menționat că utilizarea acestor linii este facilitată și de existența pe intrări a histerezisului intern din circuitul **SIO**.

De asemenea, dacă este necesară completarea canalelor seriale cu toate semnalele pentru modem, se poate conecta în mod solidar mecanic o placă auxiliară cu convertoare de nivel, conectoarele **SERIAL-A**, **SERIAL-B** și **TIMING** fiind montate pe aceeași latură a plăcii de cablaj imprimat a nucleului de dezvoltare (**anexa 2.**).

2.5. Porturile paralele.

Utilitatea unui microsistem, fie el cu microrocesor sau cu microcontroller, constă în primul rând în capacitatea de a asigura controlul unui număr mare de dispozitive (pentru diverse comenzi sau acționări electrice) prin luarea unor decizii în urma procesării datelor obținute de la un număr mare de senzori și traductoare, sau de la alte echipamente de calcul. Din acest motiv, existența unor porturi de intrare/ieșire de tip paralel, dar în care biții pot fi prelucrați distinct, este esențială.

Există o serie diversă de metode și circuite integrate pe o scară mai largă sau mai restrânsă cu ajutorul cărora se poate realiza acest deziderat pentru un microprocesor **[3.]** (în cazul unui microcontroller, se pune doar problema unei eventuale extinderi a resurselor deja disponibile "on-chip"). În cazul microprocesorului **Z80**, rămân două posibilități mai avantajoase: montarea unui circuit periferic de tip **I8255**, sau a unui circuit specific familiei microprocesorului, adică **Z80-PIO**.

Circuitul **I8255** dispune de 3 porturi paralele de 8 biți, care pot fi folosite și la transfer paralel de date cu protocol hardware, dar prezintă 3 inconveniente majore: în primul rând biții unui port nu pot fi utilizați în mod distinct ca intrări sau ieșiri, trebuind "grupați" în seturi de 4 (numai pe portul C) sau chiar de 8 biți (pe celelalte porturi A și B), fiecare grup fiind la un moment dat numai intrare sau ieșire.

Al doilea inconvenient apare numai în cazul sistemelor cu microprocesor **Z80** și constă în necesitatea montării unui controller de întreruperi distinct (în cazul altor microprocesoare, cum ar fi, de exemplu, **I8086**, acest lucru era oricum absolut necesar **[10.]**), sau renunțarea la lucrul cu întreruperi de la porturile paralele. Cea de-a treia problemă a circuitului **I8255** este ***resetarea tuturor biților care sunt ieșiri ori de câte ori se schimbă modul de lucru numai pe unul din porturi***, neputându-se schimba sensul unor linii pe timpul cât se folosesc ieșiri pentru controlul sau configurarea unui echipament **[20.]**.

Circuitul **Z80-PIO** elimină toate aceste dezavantaje, motiv pentru care am realizat interfața paralelă a sistemului montând acest circuit, chiar dacă nu dispune decât de 2 porturi.

Consider că o scurtă prezentare a posibilităților circuitului **[2.]** este necesară, atât pentru justificarea flexibilității sistemului proiectat, cât și din punctul de vedere al unui eventual utilizator care folosește sistemul într-o aplicație. De asemenea, cu acestă ocazie doresc să sugerez și câteva posibile utilizări originale pentru semnalele de protocol ("handshake") ale circuitului.

Porturile circuitului **Z80-PIO** pot fi programate să lucreze în patru moduri distincte: "*modul 0*" = ieșire pe octet, "*modul 1*" = intrare pe octet, "*modul 2*" = magistrală de transfer bidirecțională de 8 biți (numai pentru portul A) și "*modul 3*" = intrare/ieșire pe bit. Există posibilitatea controlului hardware a transferului, folosind câte două semnale separate pentru fiecare port: ieșirea **READY** (**ARDY**, respectiv **BRDY**), activă *HIGH* și intrarea **/STROBE** (**/ASTB**, respectiv **/BSTB**), activă *LOW*, pentru confirmare (cu rol de **/ACK**, dar puțin diferit).

În modul ieșire octet, **RDY** devine "1" când s-au înscris datele, iar frontul pozitiv al **/STB** semnalează printr-o întrerupere că dispozitivul conectat la port a citit datele, dezactivând totodată și semnalul **RDY** până la o nouă înscriere a unui octet de către unitatea centrală. Pe durata cât semnalul **/STB** corespunzător portului este în "1" logic, semnalul **RDY** repetă de fapt, cu un decalaj de o perioadă de tact sistem, semnalul de scriere **/WR** generat la o scriere **I/O** din partea procesorului.

În modul intrare pe 8 biți, **/STB** permite dispozitivului conectat la port să eșantioneze datele în "latch"-ul de intrare al portului prin frontul pozitiv, cu generarea unei întreruperi către unitatea centrală și dezactivarea hardware a semnalului **RDY**, care devine din nou "1" abia după ce microprocesorul a citit datele din port. Dacă semnalul **/STB** era "0" logic pe perioada transferului, semnalul **RDY** repetă cu un decalaj de un tact semnalul de citire **/RD** primit de la procesor.

Pentru conectarea paralelă a două sisteme cu microprocesoare **Z80**, dacă fiecare din cele două sisteme conține câte un circuit **PIO**, există o posibilitate interesantă care rezultă din caracteristicile acestor semnale de protocol. Se poate conecta un port **PIO** programat ca ieșire dintr-un sistem – cu un port **PIO** programat ca intrare din celălalt sistem, cu legarea încrucișată a liniilor de protocol (**RDY**(1) la **/STB**(2), respectiv **/STB**(1) la **RDY**(2)). Nu este necesară inversarea nivelului activ al semnalelor **RDY** corespunzătoare porturilor celor două circuite, deoarece contează numai frontul lor pozitiv în strobarea datelor. Într-o astfel de situație, transferul de date între cele două sisteme se face paralel, prin protocol hardware de mare viteză (folosind numai întreruperi).

În modul bidirecțional, numai portul A poate fi transformat într-o magistrală de date exterioară, folosind pentru comanda ieșirii liniile de protocol ale portului A, iar pentru intrare

semnalele de protocol ale portului B (**fig. 2.5.1**). Modul de lucru în intrare sau în ieșire este asemănător cu modurile anterioare, cu deosebirea că datele sunt plasate pe magistrala portului A numai pe durata cât **/ASTB** este activ, frontul pozitiv al acestui semnal putând fi folosit pentru memorarea datelor în dispozitivul conectat la port (/**ASTB** lucrează de fapt ca un semnal **/RD**, de citire din memorie sau din **I/O** provenit de la un procesor, iar **/BSTB** are rolul complementar, de semnal **/WR**, provenit tot de la o unitate centrală externă sistemului).

Am menționat aceste detalii de funcționare deoarece **PIO** ***poate fi astfel intercalat direct ca o interfață paralelă între două sisteme*** (cu microprocesoare sau cu microcontrollere), conectând portul A la magistrala de date a celuilalt sistem. Într-o astfel de conectare, celelalte două semnale, **ARDY** și **BRDY**, devin semnale de disponibilitate a perifericului sau memoriei în sistemul care nu conține circuitul (de tip **READY** sau **/WAIT**), sau pot fi conectate la intrări de întrerupere de tipul ***INTn***, active *HIGH*, pentru a asigura un transfer de mare viteză între sisteme, controlat complet hardware prin întreruperi.

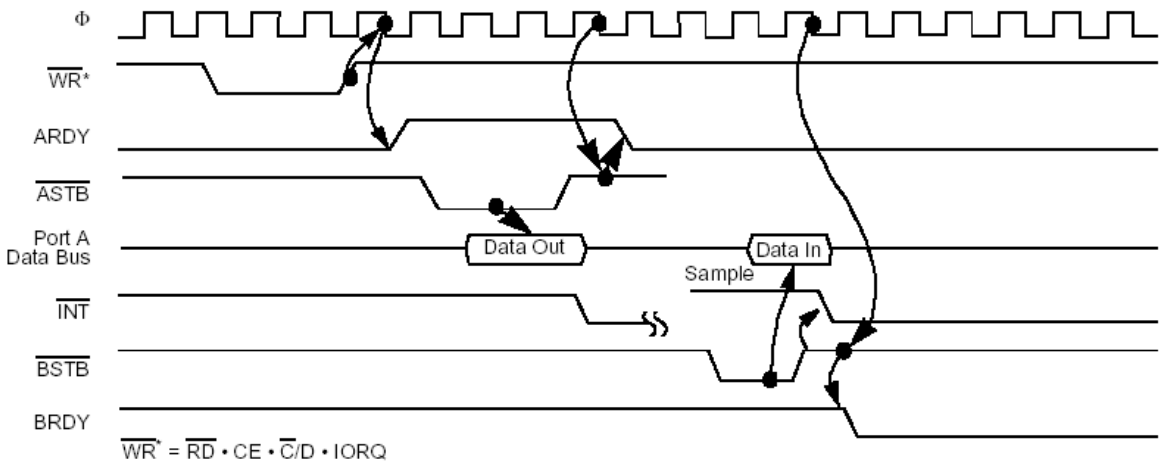


Fig. 2.5.1.

În <u>*modul 3*</u>, oricare bit din port poate fi linie de intrare sau de ieșire, cu posibilitatea de citire a biților care sunt ieșiri și a programării unei logici cablate pe biții de intrare care să valideze o condiție de întrerupere. Menționez că lucrul în acest mod este foarte asemănător porturilor microcontrollerului **PIC 16F84**, existând un registru prin înscrierea căruia se stabilește direcția pinului din port **[11.]**, dar posibilitatea programării nivelului activ și al tipului logicii (**AND** sau **OR**) care trebuie să o îndeplinească configurația de biți pentru a genera întrerupere este mai avansată la **PIO**.

Schema conectării circuitului de interfață paralelă în sistem este cea din **figura 2.5.2.**

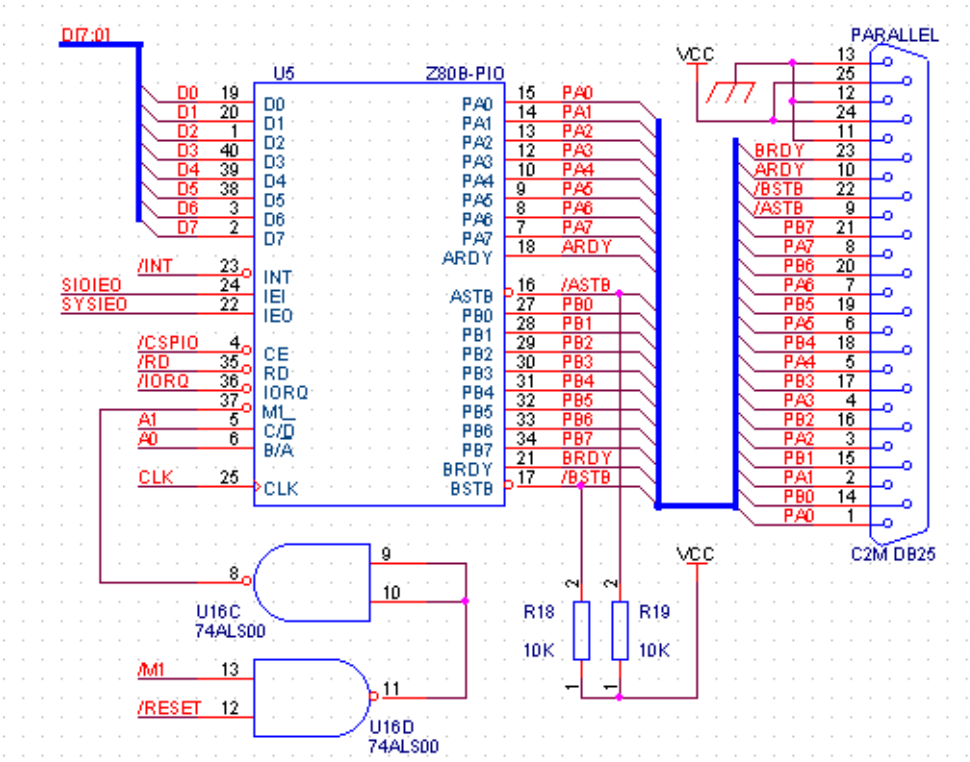


Fig. 2.5.2.

După cum se observă din schemă, circuitul este conectat la magistrala sistemului într-un mod analog cu celelalte circuite periferice prezentate anterior, fiind ultimul circuit de pe placă în lanțul de întreruperi serial - distribuit specific familiei de componente **Z80**. Semnalul **IEO** de la **PIO** l-am scos afară, la un conector pentru extensia sistemului (sub denumirea **SYSIEO**). Există însă o particularitate de conectare specifică circuitului și anume logica suplimentară necesară pentru introducerea semnalului de resetare, **/RESET**.

Datorită limitării la 40 de pini pentru capsula clasică (*PDIP = "Plastic Dual Inline Package"*), semnalul de resetare a circuitului a fost omis, această funcție obținându-se prin activarea singulară a semnalului **/M1**, neânsoțit de **/IORQ** sau de **/RD**, această combinație fiind invalidă în ciclurile microprocesorului **Z80** (vezi **&2.1.**). Pentru ca această activare să se producă odată cu resetarea generală a sistemului, am montat cele două porți **NAND** care funcționează de fapt ca o singură poartă **OR** pentru semnale active *LOW* (am montat 2 **NAND**-uri în loc de un singur **AND** din considerente de minimizare și utilizare completă a capsulelor de circuite integrate folosite, în capsula **LS 00** mai găsindu-se încă două porți **NAND**, care erau necesare în altă parte).

Liniile porturilor, împreună cu semnalele de protocol și două trasee de alimentare sunt scoase la conectorul "PARALLEL" de tip **DB25-M** al sistemului, pinii fiind alocați, de această dată, conform unui "standard" propriu, deoarece am mai realizat anterior câteva montaje

(programatoare de **EPROM**-uri, interfețe paralele pentru microcalculatoare cu **Z80**, module paralele pentru comenzi) și cabluri paralele de interconectare, care respectă toate acest asa-zis "standard", interfața paralelă cu **PIO** neputând fi "constrânsă", oricum, standardului paralel **LPT** de la **PC**-uri (semnalele de protocol sunt diferite, iar porturile au un grad mai mare de flexibilitate).

De asemenea, ordinea pinilor în conector simplifică interconectarea circuitului **PIO** cu mufa paralelă pe cablajul imprimat, având totodată și o semnificație mnemotehnică. Tipul conectorului ("male"), asigură imposibilitatea conectării inversate a cablului nesimetric atunci când se face legătura paralelă cu un calculator **PC**, care are conector de tip "female". (vezi **&4.**).

Pentru ca cele două linii de confirmare a transferului (**/ASTB**, **/BSTB**) să nu genereze întreruperi necontrolate, am montat rezistoare de "pull-up" spre Vcc, de valoare relativ mare, astfel încât să poată fi forțate *LOW* prin ieșiri MOS-LSI, cu "fan-out" slab. Acest lucru implică totuși atenție la imposibilitatea citirii unui port programat să funcționeze în mod intrare fără "șraparea" la masă a liniei **/STB** corespunzătoare. Pentru intrări simple, fără protocol, se recomandă, oricum, folosirea *modului 3* de lucru al circuitului **PIO-Z80**.

2.6. Interfața I2C realizată hardware.

Interfețele prezentate până acum erau oarecum clasice, ținând seama și de faptul că sunt realizate în jurul unor circuite din familia procesorului **Z80**. Aceste circuite se întâlnesc destul de frecvent și în schema altor sisteme de dezvoltare, chiar dacă nu în exact aceeași conectare. Pentru a spori gradul de versatilitate și de actualitate al sistemului, asigurarea unei interfețe pentru "bus"-ul serial standardizat – **I2C**, ar fi deosebit de utilă, deoarece în prezent există un număr mare de dispozitive periferice care se pot conecta astfel, iar legătura cu și între acestea se poate face folosind numai două linii (**SCL** = "Serial CLock" și **SDA** = "Serial DAta") **[11.]**.

Ca exemple de circuite ce se pot conecta pe magistrala serială **I2C**, pot fi amintite:

- memorii seriale de tip EEPROM (cu ștergere electrică) de diferite dimensiuni: (**24C00 ÷ 24C512**) – 5V, (**24LC00 ÷ 24 LC512**) – (2.5V÷6V);
- ceas de timp real cu calendar și eventual cu memorie RAM adițională (**PCF 8563**, **PCF 8573**, **PCF 8583** – RAM, **PCF 8593**);
- circuite pentru expandarea porturilor paralele (**PCF 8574** – 8 biți, **PCF 8575** – 16 biți);

– driver pentru display-uri numerice cu LED-uri care asigură atât amplificarea în curent (programabilă) cât și explorarea digiților (**SAA 1064**);
– senzori și traductoare pentru mărimi electrice și neelectrice cu convertoare analog-numerice și interfață serială înglobate;
– diferite circuite analogice cu comandă numerică prin interfață **I2C** (amplificatoare, corectoare, "tuner"-e) utilizate în aplicații de audiofrecvență sau radio-tv;

Interfața **I2C** poate fi realizată prin software, folosind 2 biți ai unui port paralel, dar acest mod de realizare are dezavantajul că solicită atenția permanentă a procesorului (la generarea ceasului serial, la serializarea, deserializarea și transferul biților dintr-un octet, la transmiterea și recunoașterea adreselor), lucrând și fără întreruperi. Se observă că încărcarea procesorului în acestă situație ar fi destul de mare, iar sistemul n-ar mai îndeplini condiția de sistem de dezvoltatre pentru control în timp real (calculator de proces). De aceea, orice degrevare a unității centrale de una din sarcinile amintite este bine venită.

De altfel, la majoritatea microcontrollerelor actuale care au interfață **I2C**, aceasta are o parte mai mică sau mai mare realizată prin hardware. Pentru sistemul proiectat, care se vrea un "calculator de control" mai puternic, implementarea completă prin hardware a interfeței **I2C** devine aproape o necesitate.

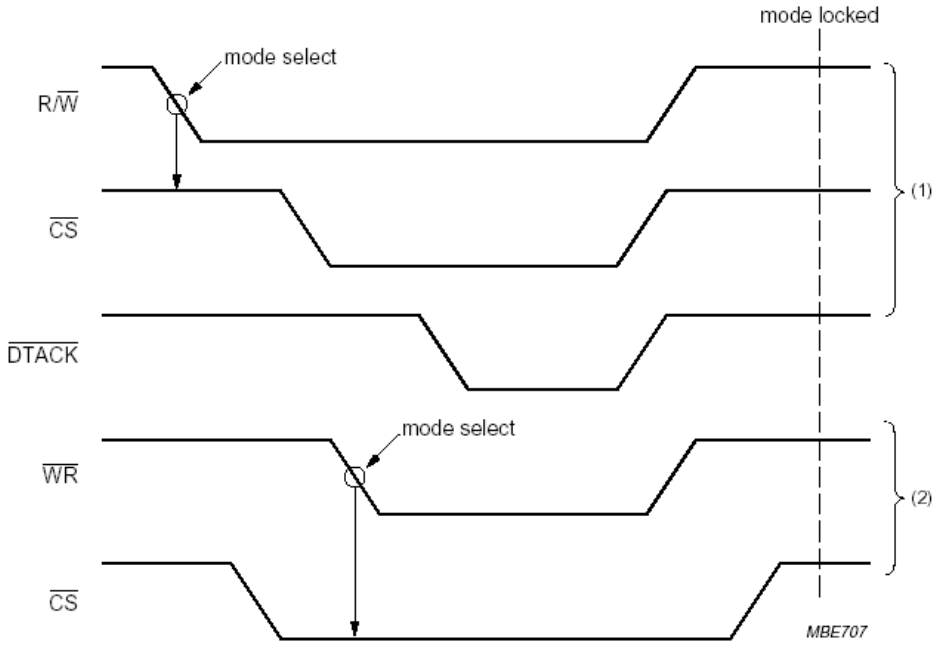
Din aceste motive, am proiectat interfața **I2C** prin hardware, cu ajutorul unui controller **PCF 8584 [18.]**. Acest circuit servește drept interfață între magistralele paralele de 8 biți ale celor mai comune microprocesoare și microcontrollere (**I8080**, **M6800**, **MCS-51**) și "bus"-ul serial **I2C**, putând fi adaptat și la magistralele altor microprocesoare (**Z80**, **I8086**, **M68000**), utilizând o serie de circuite suplimentare. Controllerul implementează hardware toate funcțiile necesare intrefeței, putând lucra atât ca "slave" cât și ca "master" pe bus, permițând sistemelor cu magistrale paralele să comunice bidirecțional cu "bus"-ul **I2C**, direct pe 8 biți, folosind un protocol de tip "pooling", sau bazat pe întreruperi.

Ca facilități interesante pentru magistrala **I2C**, pot fi amintite posibilitatea de lucru în sisteme multi-master, posibilitatea de monitorizare a traficului pe bus și modul de lucru la distanță ("long-distance") care folosește pentru comunicație 4 fire, față de cele două fire utilizate în mod standard (liniile de ceas și de date nu mai sunt bidirecționale, existând câte o intrare și ieșire pentru fiecare). În acest mod se pot monta convenabil translatoare de linie pentru comunicația la distanță între două sau mai multe sisteme paralele prin magistrala **I2C**.

De asemenea, circuitul poate furniza ceas serial programabil prin câteva trepte de divizare, iar intrările de pe liniile seriale sunt trecute prin filtre digitale pentru eliminarea perturbațiilor, atât în modul clasic cât și în modul pentru distanță lungă.

După cum am amintit, circuitul nu este direct compatibil cu microprocesorul **Z80**, fiind necesare o serie de adaptări. În primul rând, trebuie selectat modul de lucru compatibil ***Intel***, pentru magistrală de tip **8080**, pentru ca apoi să se poată folosi adaptările "clasice" ale circuitelor periferice din familia **I82xx** la microprocesorul **Z80**. În acest sens, **PCF 8584** permite adaptarea automată la unul din cele două tipuri de magistrale mai des întâlnite (**I8080** sau **M6800**), prin detectarea primei secvențe de semnale **"/CS - /WR"** ce apare după resetare (**fig. 2.6.1.**) **[18.]**.

Acest concept se folosește de faptul că semnalul de scriere este comun ambelor tipuri de magistrale. Dacă se detectează o tranziție *HIGH => LOW* pe controlul de scriere **/WR** cât timp semnalul **/CS** este inactiv (*HIGH*), atunci se selectează modul de lucru **M6800**. În caz contrar, circuitul rămâne în modul **I80xx**, care este modul de lucru implicit, după resetare. Așadar, după cum se observă din diagramele de timp, pentru a păstra modul de lucru **I8080**, primul front negativ al semnalului **/WR** care apare la scrierea unor date în circuit trebuie să "prindă" circuitul deja selectat anterior (**/CS** = "0" logic). După prima scriere, modul de lucru este blocat intern, până la o nouă resetare, iar semnalele care diferă între cele două tipuri de magistrale sunt setate corespunzător, circuitul lucrând normal.



(1) Bus timing; 68000 mode write cycle.
(2) Bus timing; 80XX mode.

Fig. 2.6.1.

Problema care apare la sistemele ce utilizează microprocesorul **Z80**, este aceea că semnalele **/WR** și **/IORQ** apar aproape simultan în ciclul de scriere **I/O** (vezi **&2.1.** / **fig. 2.1.7.**), semnalul **/IORQ** împreună cu adresele de port corespunzătoare fiind apoi folosite la generarea

/CS pentru **PCF 8584**. Din acest motiv, întârzierea care apare la propagarea prin decodorul de adrese **I/O**, conduce la apariția frontului negativ pe semnalul de scriere înaintea selectării circuitului, trecând controllerul **I2C** în modul de magistrală ***Motorola***. De asemnea, micile variații în momentele de apariție ale celor două semnale de la microprocesor, variații care pot apărea datorită lungimii diferite a traseelor pe cablaj sau a încărcării diferite a liniilor, pot duce la apariția unui hazard nedorit în selectarea modului de lucru.

Pentru a elimina acest hazard și pentru a permite setarea sigură a modului de lucru pe magistrale, am folosit un circuit de întârziere sincron a semnalului de scriere provenit de la microprocesor, o întârziere asincronă cu ajutorul unui circuit RC fiind nesigură. Schema blocului de interfață **I2C** se găsește în **figura 2.6.2.**

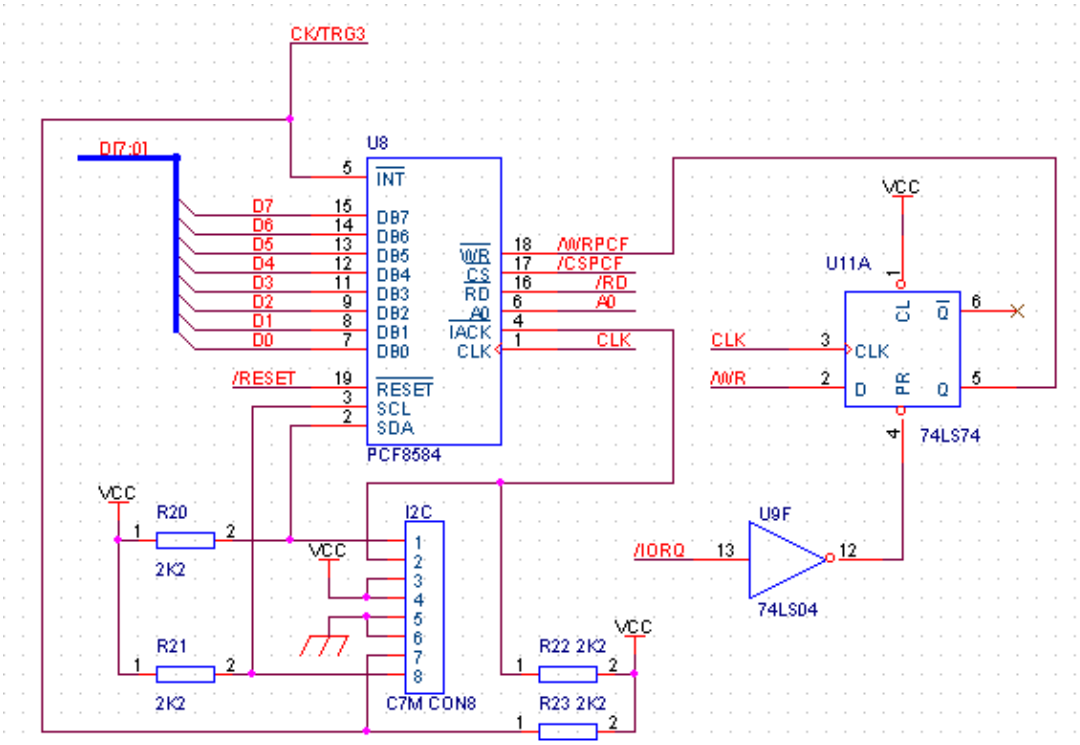


Fig. 2.6.2.

Din studiul datelor de catalog pentru microprocesorului **Z80B [17.]**, se observă că frontul negativ pe **/IORQ**, front care declanșează decodificarea selecției perifericelor (adresele fiind deja stabile), apare în perioada de tact T2, cu o întârziere de maxim ***65ns*** față de sfârșitul frontului pozitiv al tactului T2, întârzierea față de același front a semnalului de scriere **/WR** fiind de maximum ***60ns***. Deci **/WR** apare chiar cu ***5ns*** înainea lui **/IORQ**, fiind cu atât mai mult necesară întârzierea pentru semnalul de scriere în controllerul **I2C**, **/WRPCF**.

Pentru realizarea acestei întârzieri, am sincronizat semnalul **/WR** cu frontul pozitiv al ceasului - sistem, cu ajutorul unui bistabil de tip "D" cu tranziție pe front pozitiv (**LS 74**). La

frecvența de **6.144 MHz** frontul pozitiv următor apare la aproximativ ***163ns*** după frontul pozitiv al lui T2, rezultând pentru **/WRPCF** o latență de: ***163ns – 60ns = 103 ns*** plus timpul de propagare prin bistabil, timp suficient pentru orice decodificator de adrese să genereze semnalul de selecție **/CSPCF** înainte (în cazul decodării cu **LS 138** după schema folosită în sistem timpul de propagare este de maxim ***33ns***, la care se adaugă cele ***5ns*** întârziere datorate lui **/IORQ**, rezultând un timp total de întârziere de aproximativ: ***103ns – 33ns – 5 ns = 65 ns***, mult mai mare decât minimul de ***20ns*** specificat în foaia de catalog pentru circuitul **PCF 8584**).

De asemenea, beneficiind din plin de perioada suplimentară Tw introdusă în mod automat de microprocesorul **Z80** în ciclul de intrare/ieșire, întârzierea semnalului de scriere nu este nici prea mare, astfel încât scrierea datelor să fie realizată cu succes, în timp util (lățimea palierului *LOW* al semnalului de scriere la **Z80B** este de aproximativ ***580ns***, rezultând un timp de scriere în controllerul **I2C** de ***580ns – 103 ns = 477ns*** dacă semnalul de scriere în controller s-ar dezactiva odată cu semnalul **/WR** de la procesor, timp care este mult mai mare decât minimul de catalog cu valoarea ***230ns***).

Problema care mai rămâne este eliminarea acestei întârzieri la tranziția *LOW => HIGH* a semnalului **/WR** la terminarea ciclului de scriere în **PCF**, pentru a nu perturba eșantionarea datelor în controller, care are loc odată cu frontul pozitiv al semnalului de scriere. În acest sens, foaia de catalog **[18.]** precizează că dezactivarea lui **/WRPCF** trebuie să se facă înainte de **/CSPCF** cu cel puțin ***0ns***, adică semnalele se pot dezactiva simultan, dar nu în ordine inversă. Pentru evitarea repetării sincronizării cu frontul pozitiv al tactului următor la dezactivarea semnalului de scriere, am forțat în "1" logic ieșirea bistabilului printr-una din intrările sale asicrone (și anume intrarea /PRESET – activă pe "0" logic) odată cu frontul pozitiv al semnalului **/IORQ**, folosind un inversor **LS 04**.

Bistabilul este forțat astfel în "1" logic pe toată durată cât **/IORQ** rămâne inactiv, schimbarea stării bistabilului fiind permisă din nou numai după reactivarea lui **/IORQ**. Bistabilul care realizează această sincronizare are o comutare sigură, de la comutarea pe *LOW* a semnalului **/IORQ** până la următorul front pozitiv al tactului timpii de "set-up", de menținere și de propagare la comutare asincronă fiind suficienți.

După setarea modului de lucru cu magistrală de tip **I8080**, compatibilizarea acesteia cu magistralele microprocesorului **Z80** se realizează în modul clasic folosit la conectarea circuitelor din familia **I82xx**, prin introducerea în decodificatorul de adrese **I/O** a semnalului suplimentar **/IORQ**, după cum am amintit în paragraful **&2.1.**

Altă problemă care apare la conectarea controllerului **I2C** este integrarea sa în sistemul de întreruperi al microprocesorului **Z80**. În acest sens, trebuie remarcat că utilizarea posibilității de depunere pe magistrală a unui vector de întrerupere propriu (păstrat într-un registru intern

special) în ciclul de achitare a întreruperii, după primirea semnalului **/IACK** pe zero logic, nu rezolvă problema, deoarece trebuie realizată atât arbitrarea priorităților (cu invalidare atunci când un periferic mai prioritar se află în cursul achitării), cât și generarea semnalului **/IACK** din decodarea combinației de semnale **/M1** = ”0” și **/IORQ** = ”0” care apare numai la achitarea unei întreruperi mascabile ce a fost semnalată anterior către unitatea centrală prin activarea liniei **/INT**.

Într-o variantă anterioară a schemei sistemului de dezvoltare, am realizat conectarea în lanțul de întreruperi în acest fel, folosind semnalele **IEI** și **IEO** care permit propagarea serie a priorităților între circuitele periferice din familia **Z80** și o logică adițională cu porți pentru blocarea cererii de întrerupere proprii controllerului, invalidarea circuitelor mai puțin prioritare, și generarea semnalului **/IACK**.

După ce am încercat alocarea optimă a porților per capsulă, am constatat că această rezolvare nu este avantajoasă deoarece mărea semnificativ numărul de circuite integrate și de interconexiuni între acestea pe cablaj, încălcând totodată și principiul după care am proiectat sistemul: “Minimum necesar de circuite integrate, în special minimum de circuite cu integrare pe scară redusă (**SSI**) și utilizarea completă a porților sau bistabililor din capsulele acestora”.

De aceea, am renunțat relativ repede la acea variantă, realizând intercalarea în sistemul de întreruperi prin intermediul canalului 3 al circuitului **CTC**, conectând cererea **/INT** la intrarea **CK/TRG3** a acestuia (vezi **&2.3.**). Această abordare nu este numai mult mai economică, ci și mult mai flexibilă, deoarece atunci când nu se lucrează cu interfața **I2C** prin întreruperi canalul respectiv al circuitului “timer-counter” poate fi utilizat în alte scopuri (ceea ce nu se putea spune despre logica cu porți ...).

Intrarea **/IACK** de confirmare a acceptării cererii de întrerupere trebuie inactivată (am montat un rezistor la Vcc), pentru a nu plasa pe magistrală vectorul de întrerupere intern, ci vectorul circuitului **CTC**, care lucrează drept controller de întreruperi. Revenirea ieșirii **/INT** în starea inactivă (*HIGH*) se va face în acest caz prin software.

Conectorul pentru magistrala **I2C** este de tipul ***2 secțiuni X 4 pini***, având scoase și liniile de alimentare pe fiecare secțiune, intercalate între semnalele utile pentru reducerea diafoniei (o conectare standard, frecvent întâlnită). Pe prima secțiune sunt conectate semnalele clasice **SCL** și **SDA**, iar pe cealaltă secțiune mai sunt scoase încă două fire, **/INT** și **/IACK**, toate aceste patru semnale fiind intrări sau ieșiri de tipul cu drenă în gol, având montate rezistoare externe la Vcc, care “trag” liniile de intrare intr-o stare bine definită și refac nivelul “1” logic al ieșirilor “open drain”.

Am scos la mufă aceste semnale în scopul de a putea lucra cu controllerul și în modul "long-distance", mod în care nu se pot folosi întreruperi, semnalele modificându-și semnificațiile în felul următor **[18.]**:

modul I2C standard		modul "long distance"
SCL	=>	**SCL IN**
SDA	=>	**SDA OUT**
/INT	=>	**SCL OUT**
/IACK	=>	**SDA IN**

Se observă că rezistoarele necesare liniilor **/INT** și **/IACK** pentru lucrul cu întreruperi devin acum rezistoare de "pull-up" absolut necesare pentru magistrala **I2C**.

De asemenea, dacă se lucrează în modul standard, dar fără întreruperi, prezența la unul din pinii conectorului **C7M** a intrării de declanșare/numărare **CK/TRG3** este foarte utilă, ea putând servi drept linie de întrerupere pentru circuite **I2C** care au această facilitate (cerere de întrerupere separată de cea prin magistrala **I2C**) sau poate fi utilizată în alte scopuri (de exemplu, pentru conectarea comodă a unui senzor în infraroșu).

2.7. Conectorul pentru LCD compatibil Seiko.

Pentru afișarea unor date, parametri, sau comenzi de lucru în mod autonom, fără necesitatea conectării la un calculator-gazdă prin interfața serială sau paralelă, am dotat sistemul cu posibilitatea montării printr-un conector special a unui dispozitiv modern de afișare cu cristale lichide (**LCD** = "Liquid Crystal Display module"), de tipul alfanumeric (matrici de puncte), cu controller inlclus de reâmprospătare a "display"-ului din memoria locală, neavând astfel nevoie de baleierea ecranului pe coloane realizată prin program (cu folosirea întreruperilor periodice de la un ceas de "refresh").

Cele mai răspândite afișaje de acest fel sunt bazate pe controllerul **HD 44780** după un standard compatibil Seiko **[11.]**. Un astfel de dispozitiv este văzut din exterior ca un periferic paralel, cu două tipuri de registre de 8 biți: registrul de instrucțiuni/stare și registrul de date, selecția între ele realizându-se folosind semnalul **RS** = "Register Select".

Prin scrierea în registrul de instrucțiuni se transmit comenzile necesare pentru controlul afișajului (cum ar fi ștergerea ecranului, poziționarea cursorului, stabilirea adresei de început

pentru transferul datelor), iar prin citirea registrului de stare se poate afla disponibilitatea controllerului de a primi noi instrucțiuni (se citește indicatorul **BF** = "Busy Flag" care este zero dacă controllerul nu este ocupat cu execuția unei operații interne) și contorul de adresă curentă pentru transferul de date.

Registrul de date stochează temporar toate datele transferate în sau din memoria-ecran (**DDRAM** = "Display Data RAM) sau memoria generatorului de caractere (**CGRAM** = "Character Generator RAM").

Citirea / scrierea informațiilor este controlată de semnalul de selecție **E** = "Enable", activ *HIGH*, direcția transferului fiind stabilită prin semnalul $R/\overline{W}$. Transferul cu unitatea centrală se poate face pe 8 biți, printr-un singur puls de activare pe linia **E**, sau folosind două cicluri de magistrală succesive pe câte 4 biți marcate prin două stroburi de selectare, în funcție de modul programat imediat după inițializare. În **figura 2.7.1.** se pot urmări diagramele de timp în cazul lucrului pe 8 biți.

Interfața standard Seiko mai conține două linii de alimentare și un pin notat VLC la care se aplică o tensiune pozitivă între GND și VCC=5V, pentru reglajul contrastului afișajului (această linie este de fapt borna de alimentare cu polaritate negativă pentru afișajul propriu-zis cu cristale lichide) **[11.]**.

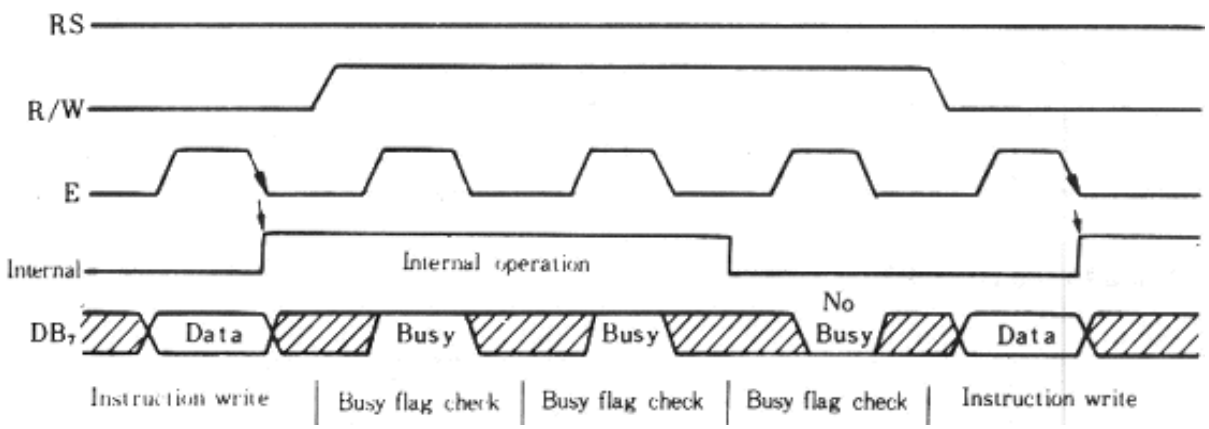


Fig. 2.7.1.

Există două moduri principale de conectare a afișorului **LCD** într-un sistem cu microprocesor sau microcontroller. Prima metodă folosește ***conectarea pe porturile paralele*** și este mult folosită mai ales în cazul microcontrollerelor fără acces extern la magistrale. Totuși, ***metoda este ineficientă***, deoarece pe lângă consumul de biți din porturi, mai necesită generarea prin program a semnalelor de selecție a registrelor, de control al sensului transferului, precum și formarea software a semnalului de activare – **E** (după plasarea pe biții de port corespunzători

semnalelor **RS** și $R/\overline{W}$ a valorilor dorite, se dă mai întâi un front pozitiv pe linia **E**, se transferă datele, apoi se dă un frontul negativ pe aceeași linie).

Această metodă este destul de des folosită și în cazul conectării altor periferice decât **LCD**-uri, chiar și la sisteme cu microcontroller cu acces extern la magistrale sau cu microprocesor, conducând la scăderea semnificativă a vitezei (în cazul **LCD** acest lucru nu deranjează, dispozitivul fiind lent), dar mai ales ***complicând inutil partea software de servire a perifericului***.

Cea de-a doua metodă, mult mai eficientă, constă în ***conectarea controllerului* LCD *direct la magistralele microprocesorului***, semnalul **E** generându-se în mod automat prin selecția dispozitivului ca periferic (sau ca memorie – în cazul mapării pe memorie), celelalte două semnale provenind direct de la procesor (se poate conecta atât **RS** cât și $R/\overline{W}$ la pinii de adresă – în acest caz citirea se face pe un port și scrierea pe altul, sau se conectează numai **RS** la un pin de adresă, folosind pentru controlul $R/\overline{W}$ linia **/WR** a microprocesorului).

Am adoptat a doua variantă de interfațare, prevăzând sistemul de dezvoltare cu un conector special pentru afișor **LCD** care permite accesul acestuia direct la magistrala de date, pe toți cei 8 biți. După cum se observă și din schemă (**fig. 2.7.2.**), selecția registrului de lucru (instrucțiuni/stare sau date) se face prin bitul **A0** al magistralei de adrese, în timp ce direcția transferului este controlată cu linia **/WR** a lui **Z80** (nu atât pentru economisirea de porturi, cât pentru ***evitarea conflictului care ar fi putut să apară la tentativa de scriere pe portul de citire***, dacă sensul de transfer era controlat cu **A1**), pulsul de selectare (**E**) provenind de la decodorul de adrese pentru periferice (reamintesc că acest decodificator este validat și prin semnalul **/IORQ**), din inversarea liniei **/CSLCD**.

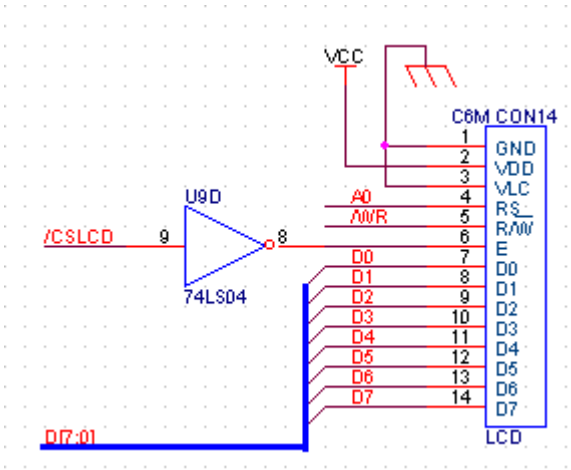


Fig. 2.7.2.

Din motive de simplitate, am renunțat la posibilitatea de reglare a contrastului pe pinul **VLC** cu un divizor rezistiv potențiometric (care ar fi avut, de altfel, și o fiabilitate destul de slabă), prin legarea la masă a liniei, deoarece nivelul optim al acestei tensiuni este, în medie, mai apropiat de masă decât de Vcc la majoritatea afișajelor. Precizez că în acest mod se asigură și un pin suplimentar de masă, util în cazul când mufa respectivă se folosește în alte scopuri. De altfel,

potențiometrul respectiv se poate înlocui cu un divizor rezistiv fix montat pe cablul de legătură sau direct pe modulul **LCD**, dacă este absolut necesar.

2.8. Blocul de comutare a alimentării bancului – RAM1 pe baterie.

Deoarece sistemul se folosește la dezvoltarea unor aplicații, este utilă posibilitatea de a păstra în memoria **RAM** o serie de ***date*** (parametri și comenzi de configurare a sistemului, variabile de lucru, tabele de întreruperi) sau chiar ***programe*** (variante de rutine în lucru, subrutine de tratare a întreruperilor) și după deconectarea tensiunii principale de alimentare.

În acest sens, dacă se prevede în exterior (nu am introdus un astfel de circuit în schemă) și un comparator rapid care să sesizeze căderea tensiunii de alimentare anunțând microprocesorul printr-o întrerupere nemascabilă (pe linia **/NMI**) că trebuie să-și salveze registrele și starea internă în memoria **RAM** astfel alimentată, programele de aplicații complexe vor putea fi reluate după realimentarea și reinițializarea sistemului exact din același punct în care au fost întrerupte, prin reâncărcarea stării și a registrelor din **RAM**-ul nonvolatil.

Pentru asigurarea alimentării de la sursa principală în condițiile în care aceasta este prezentă și comutarea rapidă pe o sursă de rezervă (baterie) în regim de avarie, schema trebuie dotată cu un circuit suplimentar de comutare a alimentării unui banc de memorie **SRAM** din sistem între cele două surse de tensiune (aici se observă și avantajul utilizării unor memorii statice în loc de memorii dinamice care ar fi trebuit reâmprospătate).

Deoarece primul banc de memorie – **RAM0**, este "expandat" cu memoria **EPROM** (vezi **&2.1.**), am preferat ca bancul celălalt – **RAM1**, să fie alimentat cu protecție. Protejarea acestui banc este mai utilă și datorită faptului că este mapat în jumătatea superioară a spațiului de memorie al microprocesorului, unde se păstrează mai convenabil rutine de sistem (**BIOS**), variabile sau tabele de întreruperi, zona inferioară rămânând pentru programe de lucru.

Un astfel de circuit de comutare a alimentării transformă de fapt acest banc de memorie **SRAM** obișnuită într-un banc special de memorie nevolatilă citește-scrie – **NVRAM** (disponibil ca modul integrat, dar la un cost mult mai mare), care asigură sistemului și o oarecare autonomie, prin păstrarea programelor cu posibilitatea de execuție a lor la pornirea sistemului, chiar în lipsa calculatorului pe care au fost puse la punct și de pe care au fost încărcate acestea.

Montarea unor simple diode pentru separarea surselor de alimentare între ele nu rezolvă problema, deoarece căderea de tensiune pe diode în regimul de conducție directă este destul de

mare (>0.7V la curenți mari), afectând în primul rând tensiunea de la baterie, care are oricum o valoare mai mică decât 5V. Din acest motiv am recurs la montajul din **figura 2.8.1.** care folosește drept comutatoare două tranzistoare PNP (Q4 și Q5), căderea de tensiune pe joncțiunea emitor-colector a unui tranzistor la saturație fiind mult mai mică, aproape neglijabilă.

Tranzistoarele Q4 și Q5 sunt comandate de ieșirile pe colector ale unui etaj de amplificare diferențial **[8.]** cu cuplaj în emitor prin rezistența comună R15, etaj realizat cu tranzistoarele NPN Q2 și Q3. Rezistorul R15 are o astfel de valoare încât curentul prin emitor să fie mic *(<1mA)*, pentru a nu consuma inutil din baterie, dar suficient pentru saturarea tranzistoarelor - serie pe alimentări $\left(> \frac{I_{RAM}}{\beta_{(Q4,\,Q5)}} \right)$. La cele două intrări ale etajului diferențial sunt conectate alimentările pentru **SRAM**, trecute prin diodele D2 și D3 care compensează căderea de tensiune pe joncțiunea emitor-bază a tranzistoarelor Q4, respectiv Q5, aducând potențialul bazei tranzistoarelor din etajul diferențial la valoarea celui din colector, pentru asigurarea unei polarizări corecte a lor pe joncțiunile colector-bază.

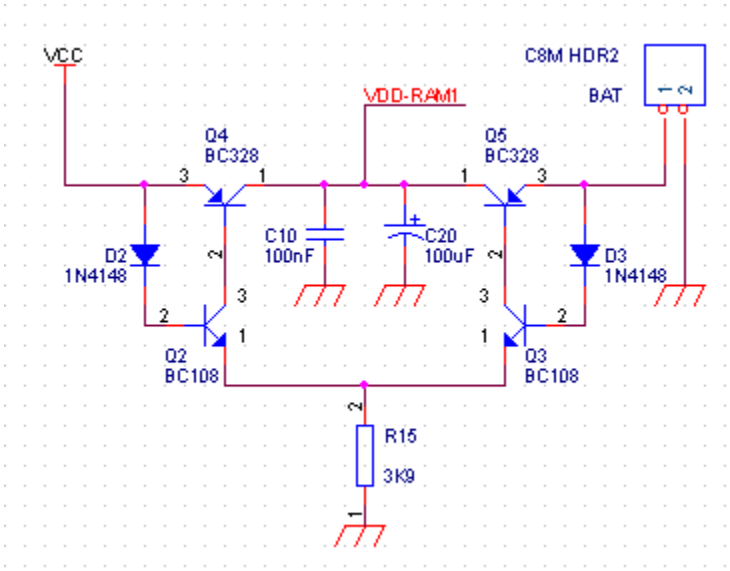


Fig. 2.8.1.

Schema are câteva proprietăți interesante, datorate etajului diferențial, care lucrează ca un comparator de tensiune, asemănător intrărilor unui amplificator operațional. Dacă tensiunile de alimentare aplicate pe intrări ar fi perfect egale și tranzistoarele – identice, curenții absorbiți din cele două surse ar fi egali. Cu cât diferența dintre valorile tensiunilor celor două surse este mai mare, cu atât montajul se dezechilibrează, absorbind mai mult curent de la sursa cu tensiunea mai mare și mai puțin de la cealaltă, astfel încât suma celor doi curenți să fie o constantă.

Pentru diferențe foarte mici ale tensiunilor de alimentare, etajul diferențial lucrează în regiunea liniară a caracteristicii de transfer **[8.]**: $|Vcc - Vbat| < Vt$, unde Vt este tensiunea termică $(25mV/20^{\circ}C)$. Pe măsură ce diferența crește, punctul de funcționare se deplasează pe caracteristica de transfer, ajungând repede în zona de saturație (pentru $|Vcc - Vbat| > 3 \bullet Vt$), când curentul trece printr-un singur tranzistor care saturează și tranzistorul regulator serie din colectorul său, conectând practic numai o singură sursă la alimentarea memoriei.

Deoarece tensiunea Vt este foarte mică, schema se dezechilibrează repede, lucrând practic ca un comutator. Totuși, datorită acestui mod de funcționare mai rezultă un eventual avantaj: montajul nu posedă o caracteristică de transfer abruptă, de tip comparator propriu-zis (care ar produce supraoscilații la comutare), trecerea de pe o sursă pe alta fiind "lină", pe măsura căderii alimentării scăzând treptat și curentul absorbit. În mod normal, tensiunea de pe bateria de siguranță este ceva mai mică decât Vcc, asigurându-se atfel un consum aproape nul de curent din baterie în prezența sursei principale de alimentare.

Condensatoarele C10 și C20 sunt chiar condensatoarele pentru decuplarea alimentării memoriei **SRAM** (unul electrolitic de capacitate mare și unul nepolarizat, fără efect inductiv). Ele asigură și un filtraj suplimentar al comutării surselor. Pentru Q2 și Q3 am folosit tranzistoare de uz general de tip **BC 108** (Ic=100mA/Vceo=20V), de joasă frecvență (variația alimentării are loc lent) și mică tensiune (alimentarea memoriei nu depășește 5V), iar pentru Q4 și Q5 tranzistoare asemănătoare, dar de putere mai mare și de tip PNP: **BC 328** (Ic=800mA/Vceo=25v) **[14.]**. Desigur, se pot folosi și alte tranzistoare, cu caracteristici mai avantajoase.

2.9. Posibilități de extensie a sistemului. Transformarea lui în calculator.

Principalul efort în proiectarea sistemului de dezvoltare cu microprocesor l-am făcut în direcția asigurării unei flexibilități cât mai mari în utilizarea resurselor disponibile "on board" și a unor posibilități cât mai mari de extindere prin montarea unor module adiționale folosind conectorii disponibili pe placă.

Pe porturile paralele ale sistemului se poate conecta în exterior o tastatură de tip matriceal pentru introducerea unor parametri de lucru sau comenzi fără a folosi comunicația cu un calculator - gazdă prin interfața serială. La citirea matricii de taste se poate folosi o metodă clasică de baleiaj prin întreruperi de la unul din "timer"-e, beneficiind la identificarea tastei apăsate și de bidirecționalitatea liniilor de port (se poate folosi un algoritm mai rapid de identificare, cu interschimbarea liniilor și coloanelor **[4.]**).

Pentru un număr mic de taste sau semnale de la anumiți senzori, aceștia nu se vor mai aranja într-o structură matriceală, conectându-se direct pe port, pentru a se putea folosi întreruperile generate direct la apariția unei combinații valide a liniior de intrare nemascate la un moment dat.

De asemenea, se pot conecta periferice pentru comenzi analogice sau achiziții de date, prin intermediul unor convertoare numeric-analogice sau analog-numerice care pot folosi și posibilitățile de lucru cu semnale de protocol ale circiutului **PIO [3.]**.

De exemplu, se poate conecta un convertor analog-numeric pe mai mult de 8 biți folosind ambele porturi ale circuitului în modul intrare, strobate simultan la terminarea conversiei (semnalele **/ASTB** și **/BSTB** vor fi conectate împreună și comandate de ieșirea **ENDCONV** care trece pe "1" logic la terminarea ciclului de conversie generând frontul pozitiv de strobare). Impulsul de începere a conversiei (**STARTCONV**) va fi format din semnalul **RDY** al portului care va fi citit ultimul de microprocesor, folosind un formator de puls cu circuit RC și porți, sau un monostabil (realizat, de exemplu cu un circuit βE555 **[9.]**).

Semnalul de început al primei conversii va fi dat de inițializarea semnalului **RDY** corespunzător, pritr-o citire falsă a portului, după care sistemul va lucra independent, prin protocol hardware cu întreruperi. Se poate folosi o singură rutină de întrerupere care va salva în memorie datele citite de pe ambele porturi, cu incrementarea unui pointer local, declanșând prin citire și începutul unei noi conversii.

Se observă în aceste situații ***avantajul utilizării unui port paralel cu protocol hardware***, lucru imposibil în cazul unui microcontroller integrat care nu folosește circuite auxiliare. Se

simplifică astfel semnificativ programul de lucru, cu creșterea substanțială a vitezei pentru orice aplicație care foloseste porturile paralele.

Totuși, metoda principală de extindere a sistemului constă în utilizarea conectorului de extensie pentru adăugarea unor circuite sau module suplimentare (**fig. 2.9.1.**). Acest conector este de tip magistrală, având disponibile toate semnalele procesorului plus câteva semnale din sistem necesare pentru a oferi cât mai multe posibilități de extindere.

La conectarea semnalelor pe pinii conectorului, am respectat pe cât posibil standardul folosit la microcalculatorul "**TIM-S**" **[12.]**, care mi s-a părut mai complet decât varianta folosită la microcalculatoarele "**SPECTRUM**" originale **[5.]**. Semnalele microprocesorului (magistrala de date, adrese și comenzi) respectă complet acest standard, iar semnalele suplimentare sunt adăugate în locul pinilor specifici microcalculatorului "**TIM-S**" cu funcții asemănătoare, în limita posibilităților (sistemul diferă substanțial de acest microcalculator).

Am scos în plus față de magistrale, intrarea și ieșirea canalului 0 al circuitului "timer" (**TRG0**, **TO0**), intrarea și ieșirea în lanțul de priorități serie pentru întreruperi în care sunt montate circuitele din sistem (**SYSIEI**, **SYSIEO**), cele 2 ieșiri de protocol folosite ca semnale de uz general din canalul serial B (**/RTS**B, **/DTR**B), invalidarea **RAM**-ului intern superior (**/RAMD**) și 3 semnale de selectare a dispozitivelor periferice (**/CS4**, **/CS5**, **/CS6**).

Datorită prezenței acestor semnale, se pot conecta ușor în exteriorul sistemului circuite periferice integrate suplimentare, atât din din familia microprocesorului **Z80** (***atenție:*** *lanțul de întreruperi serie suportă încă un singur circuit fără introducerea unor stări de WAIT suplimentare !*), cât și din familia **I82xx**, beneficiind și de condiționarea cu **/IORQ** a semnalelor de selecție disponibile (**/CS4**, **/CS5**, **/CS6**). Pentru circuitele din familia ***Intel***, se poate folosi și **TRG0** ca intrare de întreruperi (eventual extinsă).

Utilizând ieșirea **TO0** se pot genera în exterior semnale de ceas cu frecvența programabilă, iar ieșirile de uz general **/RTS**B și **/DTR**B, se pot folosi la paginarea unor bancuri suplimentare de memorie prin invalidarea **RAM1** cu **/RAMD**, fără maparea unor porturi suplimentare.

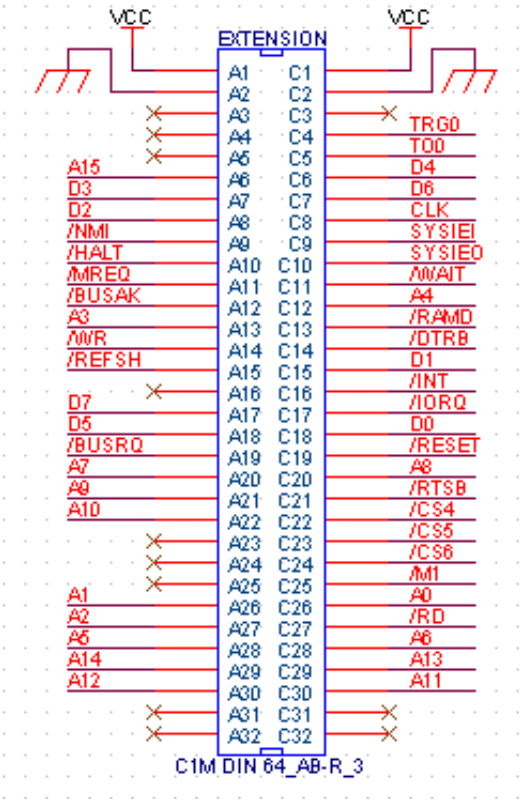


Fig. 2.9.1.

Sistemul poate fi transformat în microcalculator, prin adăugarea în exterior a unui controller video și a unei interfețe de dischetă și/sau “hard-disk”, acestea fiind singurele resurse care lipsesc.

Prin conectarea unui secvențiator video compatibil ca mod de afișare cu microcalculatoarele de tip “**SPECTRUM**” **[5.]** se poate realiza compatibilizarea în acest sens a sistemului, iar o interfață de disc permite instalarea sistemului de operare “**CP/M**”. Montarea unui hard disk cu controller integrat de tip **IDE** este facilitată și de prezența semnalelor de selecție, câte unul pentru fiecare 8 porturi (am studiat standardul **IDE** și am observat că necesită două semnale de “chip select” și 3 pini de adresă, **HDD-IDE** fiind văzut în exterior ca un set de 16 porturi intrare-ieșire).

Capitolul 3.

Realizarea și testarea sistemului.

3.1. Proiectarea și realizarea cablajului.

Pentru proiectarea cablajului, am folosit din pachetul de programe "**OrCAD 9.0**" utilitarele "***OrCAD Capture***" – pentru desenarea schemei și asocierea "footprint"-urilor componentelor și "***OrCAD Layout***" – pentru plasarea pieselor pe cablaj și "routarea" traseelor de legătură **[19.]**.

În afară de modulele prezentate, în schema sistemului (vezi **&anexa 1.**) am mai introdus un LED pentru indicarea prezenței tensiunii de alimentare și încă 9 condensatoare nepolarizate (cu capacitatea de $100nF$) și 7 condensatoare polarizate (cu capacitatea de $100\mu F$) pentru decuplarea liniilor de alimentare, pe care le-am plasat pe cablaj în imediata apropiere a circuitelor. În total, cu tot cu cele două condensatoare separate de decuplare a alimentării pentru **RAM1**, în sistem am montat (10+8) condensatoare de decuplare la un total de 16 circuite integrate (în medie, revine câte un condensator multistrat nepolarizat și unul electrolitic polarizat la fiecare două circuite).

Pentru placa de ***cablaj imprimat dublu stratificat***, am ales o dimensiune de **200mm × 160 mm**, în scopul de a dispune de o rezervă suficientă de spațiu pentru o realizare cât mai "aerisită" a montajului (loc suficient pentru trasee, trecerile de pe o parte pe alta și pentru lipirea mufelor și soclurilor circuitelor).

Plasarea componentelor pe placă în mediul "OrCAD Layout" (vezi **&anexa 2.**) am realizat-o manual, urmărind în special o distribuție cât mai uniformă, cu optimizarea intreconexiunilor dintre componente (componentele între care există mai multe interconexiuni să fie plasate în vecinătate și într-o poziție care să minimizeze numărul traseelor care se intersectează).

Pentru ***rutarea automată a cablajului*** în "Layout" (o rutare manuală fiind imposibilă datorită complexității schemei), am folosit parametrii de configurare din tabelul următor, asociați cu o lățime de ***0.35 mm*** *pentru traseele normale*, și de ***0.7 mm*** *pentru traseele de alimentare*, cu *treceri de diametrul* ***1.397 mm***. Valorile cu mai multe zecimale sunt necesare în mare parte datorită distanței de **0.01 inch = 0.254 mm** între pinii unui circuit cu capsulă **DIP** normală **[19.]**.

Visible grid:	**1.27** mm	Track to track spacing:	**0.508** mm
Detail grid:	**0.635** mm	Track to via spacing:	**0.508** mm
Place grid:	**0.635** mm	Track to pad spacing:	**0.35** mm
Routing grid:	**0.254** mm	Via to via spacing:	**1.397** mm
Via grid:	**0.254** mm	Via to pad spacing:	**1.397** mm

La rutare am avut grijă ca distanța pe cablaj dintre pinii de alimentare ai circuitelor și condensatoarele de decuplare să fie cât mai scurtă, iar lipirea terminalelor la conectori să poată fi realizată (pe cât posibil) pe o singură parte.

După ce am ajuns la o variantă convenabilă din aceste puncte de vedere (cu un număr redus, de numai ***200 de "vias"*** – treceri de pe o parte pe cealaltă), am încercat câteva variante de cablaj prin fotocopiere cu "spray fotoresist", folosind expunerea la o lampă cu vapori de mercur (care emite în zona spectrală – ultraviolet) prin folii transparente cu modelul cablajului ("măști").

Aceste variante fiind, însă, nesatisfăcătoare în cazul de față, ***am realizat cablajul manual***, prin desenarea traseelor de legătură între găuri (date în prealabil, după "măștile" de pe folii) cu ajutorul unei seringi cu ac fin (umpută cu lac de unghii) și corodare ulterioară în soluție de clorură ferică tehnică – 30%. Desigur, această metodă convine numai pentru prototip ...

Circuitul imprimat de pe suprafața superioară (fața cu piese) este prezentat în **anexa 3.**, iar circuitul de pe partea inferioară se găsește în **anexa 4.** Pentru fixarea circuitelor am folosit "socluri cu găuri", care au o anduranță mai mare în timp și pot fi lipite pe ambele suprafețe ale cablajului.

3.2. Tehnici de testare a sistemelor cu microprocesor.

Viața oricărui echipament electronic depinde în primul rând de fiabilitatea lui, iar în al doilea rând de ușurința cu care erorile de funcționare pot fi detectate și remediate. Acest ultim deziderat nu poate fi atins dacă în faza de elaborare a proiectului, aspectului de "service"-abilitate nu i s-a acordat atenția cuvenită.

Prezența unui microprocesor într-un echipament poate complica activitatea de testare, diagnosticare și depanare, putând solicita în ultimă instanță utilizarea unor aparate de test complexe, cum ar fi analizorul de stări logice, analizorul de semnături sau testorul specializat pentru acel produs. Testarea sistemelor cu microprocesor se deosebește esențial de diagnoza unor sisteme digitale clasice, prezentând dificultăți serioase cauzate de o serie de particularități specifice **[7.]**:

(1.) – Circuitele folosite au o complexitate elevată și o neregularitate de structură. Circuitele integrate **LSI** și **VLSI** folosite într-un sistem cu microprocesor conțin zeci și sute de mii de elemente logice pe un singur cip și, cu excepția memoriilor, reprezintă circuite digitale având o structură neregulată, cu foarte multe căi de transmitere a informației.

(2.) – Accesul la nodurile testate este într-o mare măsură limitat. Procesele de prelucrare a datelor în microprocesor sunt inaccesibile din exterior și practic, utilizatorul nu poate observa cauzele și consecințele directe ale defectelor. Aplicarea acțiunilor la intrările unui circuit din componența microcalculatorului pentru testarea algoritmului său de funcționare este posibilă numai cu ajutorul soft-ului, ceea ce impune elaborarea unui program complicat și o analiză detaliată a răspunsurilor căpătate.

(3.) – Majoritatea sistemelor cu microprocesor sunt racordate cu ajutorul magistralelor. Existența magistralelor care sunt folosite în comun de toate unitățile sistemului complică algoritmii de diagnosticare, întrucât defectul unei unități poate bloca sau masca funcționarea celorlalte unități racordate la magistrală. Una și aceeași unitate poate fi emițător și receptor, iar pinii respectivi în diferite momente de timp pot fi pini de intrare sau de ieșire, sau se pot afla în starea inactivă de impedanță ridicată.

(4.) – Clasa defectelor posibile este enormă. Modelul defectelor permanente ale circuitelor integrate **SSI** și **MSI** nu reflectă toate defectele posibile ale circuitelor **LSI** și **VLSI**. Gradul elevat de integrare a componentelor în aceste circuite conduce la apariția unor defecte noi, cu manifestare aleatoare datorate interacțiunii elementelor vecine pe cip, cum ar fi sensibilitatea microprocesoarelor la anumite secvențe de instrucțiuni și la frecvența semnalului de tact, sensibilitatea memoriilor **RAM** la tensiunea de alimentare și la perturbații

electromagnetice exterioare. Clasa defectelor este completată și de defectele plăcii de cablaj imprimat: cuplajul capacitiv între conductoare, inductivitatea sau rezistivitatea excesivă a unor linii, scurtcircuite sau întreruperi de trasee, etc.

(5.) – Testarea trebuie făcută în timp real. În cazul echipamentelor cu microprocesor, testarea trebuie efectuată la o frecvență prestabilită de performanțele microprocesorului, ceea ce impune performanțe deosebite mijloacelor de diagnosticare. Foarte multe defecte ale microprocesoarelor și ale memoriilor pot fi detectate numai la o frecvență înaltă de funcționare, apropiată de limita tehnologică garantată. Necesitatea testării în timp real este cauzată și de faptul că programele de test sunt foarte lungi și cer un timp îndelungat de execuție.

(6) – Verificarea circuitelor integrate LSI și VLSI este, de regulă, incompletă. Fiabilitatea funcționării unui sistem cu microprocesor este mai mare în cazul când componentele lui au trecut în prealabil un control, care, evident, ar trebui să asigure detectarea și rebutarea circuitelor integrate defecte. Totuși, o testare completă la producător este, practic, imposibilă datorită complexității obiectelor de diagnosticare, necesitând un timp foarte îndelungat. De aceea, întreprinderile producătoare de **VLSI**-uri folosesc secvențe de testare cu o lungime mult mai scurtă (de ordinul a mii de perioade ale frecvenței de tact) și 1-5% din circuitele defecte sunt declarate în ordine și expediate utilizatorului, care urmează să folosească secvențe de testare de 10-20 de ori mai lungi, dar și acestea se pot dovedi insuficiente în unele situații critice.

(7.) – În sistemele cu microprocesor softul și hardul alcătuiesc un tot unitar. Unitățile de control ale microprocesoarelor pot fi microprogramate, sistemul poate avea o serie de module hard realizate cu memorii **PROM** sau arii logice programabile (**PLA**, **FPGA**), întreaga configurare a circuitelor programabile și a unor module hard specifice este controlată de rutine **BIOS** înscrise în memoria **EPROM** a sistemului, ceea ce conduce la imposibilitatea de a indica exact unde este hardul și unde este softul, complicând suplimentar procesul de testare.

Totodată, însă, microprocesoarele posedă și o serie de particularități care simplifică procesul de depistare a defectelor din sistem, cum ar fi ***forma canonică standardizată a semnalelor*** și ***capacitatea de autodiagnosticare***. Deoarece informația binară se prezintă într-un mod standard pentru tot sistemul și datorită nucleului hardware relativ standardizat, procesul de analiză se simplifică, reducându-se la determinarea stărilor logice în anumite puncte sau linii ale magistralelor, iar măsurarea și analiza parametrilor suplimentari (amplitudinea semnalului, durata semnalului, timpii de întârziere, etc.) se efectuează numai atunci când defectele funcționale au fost depistate și este necesară localizarea lor la nivel de componente electronice.

Capacitatea microprocesoarelor de a efectua autoteste se manifestă prin faptul că imediat ce au început să funcționeze generatorul de tact și unitatea de control apare posibilitatea de a lansa în execuție programe de testare, generarea acțiunilor și analiza răspunsurilor putând fi

efectuată cu ajutorul procesorului din sistem, sub controlul unor programe stocate în memoria **ROM**. Astfel de programe permit detectarea și uneori chiar localizarea defectelor de păstrare și prelucrare a datelor în microprocesor, a defectelor de configurare sau selecție a perifericelor și memoriei, a anduranței în timp a memoriilor **EPROM**, și în special a defectelor din cipurile de memorie **RAM**.

Principalele mijloacele specifice de diagnosticare a echipamentelor cu microprocesor sunt următoarele **[7.]**: ***analizorul logic*** și ***analizorul de semnătură***, ***emulatorul în circuit*** și ***generatorul de cuvinte***.

Analizoarele logice și analizoarele de semnătură reprezintă mijloace de înregistrare, prelucrare și indicare a reacțiilor de răspuns ale obiectului diagnosticat la acțiunile stimulatoare de intrare, provenite fie de la surse interioare (în cazul diagnosticării elementelor lui), fie de la surse externe (în cazul diagnosticării obiectului în ansamblu). Analizorul logic se utilizează în principal în etapa de elaborare a hard-ului sistemului pe bază de microprocesor, iar analizorul de semnătură este mai indicat pentru întreținere și depanare decât pentru faza de elaborare a sistemului, deoarece necesită o listă de "semnături" etalon, colectate anterior. Spre deosebire de analizorul logic, care realizează pentru informația curentă o extensie sub forma unui traseu, analizorul de semnătură realizează o compresie sub forma unui singur cuvânt, denumit "semnătură".

Ideea ***analizorului logic*** a apărut din imposibilitatea proiectantului de a analiza prin intermediul unei vizualizări semnalele și fluxurile de date la funcționarea în timp real a unui sistem pe bază de microprocesor. Soluția constă în "captarea" unui anumit traseu al fluxului de date, respectiv a succesiunii în timp a unor semnale, memorarea și apoi vizualizarea lor sub o formă cât mai sugestivă, cu ajutorul "*analizorului de stări logice – software*", respectiv "*analizorului logic de timp – hardware*", care sunt cele două funcții principale ale unui analizor logic (analiza în domeniul date și analiza în domeniul timp).

Pentru analiza în domeniul timp, funcționarea analizorului este asicronă, eșantionarea făcându-se de către un tact produs în interiorul analizorului, care nu este sincronizat cu tactul sistemului și are o frecvență de cel puțin 5 ori mai mare decât acesta. În domeniul date, funcționarea este sincronă, tactul folosit de analizor fiind preluat din sistemul testat, pentru a capta numai acele stări ce reprezintă date valide.

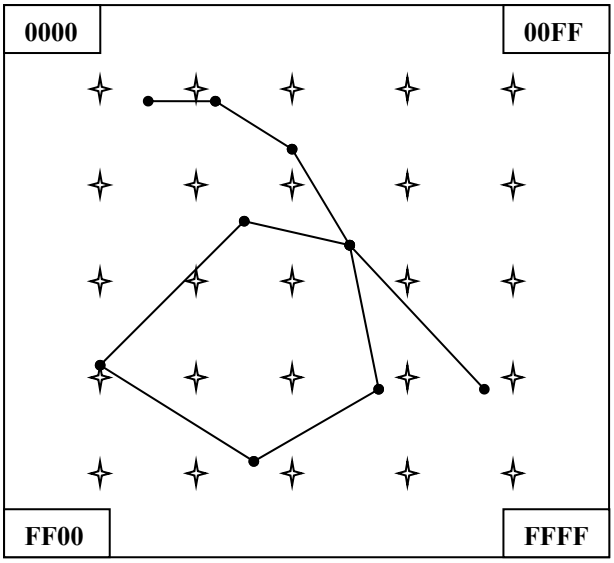


Fig. 3.2.1.

Cel mai informativ mod de reprezentare a informației colectate în memoria analizorului logic este regimul de reprezentare grafică, care permite vizualizarea succesiunilor de date cu volumul de zeci de kilobiți **[7.]**. Cel mai elocvent exemplu de reprezentare grafică este "*harta memoriei*" (**fig. 3.2.1.**).

În acest sens, pentru diagnosticarea sistemului cu microprocesor se folosește un program - test. După declanșarea rulării lui, este necesar în primul rând să ne convingem că efectuează corect adresarea și nu rămâne din cauza unui defect în altă zonă de memorie, îndeplinind un program inexistent sau stopându-se în ea. Pentru a determina acest tip de comportare a sistemului, este suficientă urmărirea stărilor liniilor magistralei de adrese, interesând doar tranzițiile de la o adresă la alta, nu și transferul de informație pe magistrala de date a sistemului testat.

Cu alte cuvinte, se reprezintă pe ecranul analizorului logic graful tranzițiilor programului de test dat de la o adresă la alta, prin asocierea unor puncte luminoase cuvintelor de adresă de până la 16 biți. Când sistemul testat folosește oricare adresă, aceasta apare în punctul respectiv al ecranului, deosebindu-se printr-o luminozitate mai mare de celelalte. Efectuând baleierea între aceste puncte, obținem un grafic numit ***harta memoriei***, pe care se poate urmări decurgerea proceselor pe magistrala de adrese.

Aparatul destinat perceperii fluxurilor complicate de date binare într-un anumit punct al obiectului diagnosticării prin formarea unui cod ce caracterizează procesele din nodurile precedente nodului dat, este ***analizorul de semnătură***. Scopul principal al analizei de semnătură este transformarea succesiunilor întinse de date binare din punctele de control ale sistemului pe bază de microprocesor într-un număr binar unic, denumit semnătură, prin folosirea unor tehnici

de compresie asemănătoare metodelor de depistare a erorilor la transmiterea datelor prin canalele de comunicație (la schimbarea doar a unui singur bit al succesiunii de date prin nodul considerat, se schimbă și semnătura).

Excitarea succesiunilor de date în nodurile de control ale sistemului este provocată, de regulă, de execuția unui test special. Semnăturile ridicate de la sistemul testat se verifică de la ieșiri spre intrări, prin comparație cu semnăturile corecte ridicate de la sistemul în stare corectă de funcționare, indicate pe schemă (într-un mod analog cu oscilogramele din sistemele analogice).

Luând în considerație că hard-ul și soft-ul unui sistem pe bază de microprocesor sunt incluse într-o buclă unică de reacție, pentru ca sistemul să se facă accesibil testării prin analiza de semnătură trebuie asigurată posibilitatea de întrerupere a acestei bucle. Întreruperea buclei de reacție în sistemele cu microprocesor monocip se efectuează prin decuplarea mecanică sau electronică a magistralei de date a sistemului de la pinii microprocesorului. Pentru aceasta, de multe ori magistrala de date a microprocesorului este separată prin rezistențe de restul magistralei sistemului, putându-se forța din exterior pe valoarea dorită (de exemplu, la microcalculatorul "**TIM-S**" **[12.]**).

Unul din cele mai simple teste folosite pentru a activiza sistemul în scopul analizei de semnătură (dar util și în cazul analizei prin alte mijloace, de exemplu cu osciloscopul) este ***regimul de rulare liberă***. Pentru asigurarea acestui regim la sistemele care nu au fost prevăzute special pentru separarea magistralei de date, se scoate microprocesorul din soclul său montat pe cablajul imprimat, între acesta și soclu introducându-se o teacă specială. De la această teacă se aplică pe intrările magistralei de date codul instrucțiunii ***NOP*** = "No OPeration" (**00H** – în cazul microprocesoarelor **I8080**, **I8085**, **Z80**, sau **90H** – pentru **I8088**, **I8086**, **X86**), concomitent întrerupându-se și magistrala de date a sistemului. Restul conexiunilor se păstrează.

După readucerea la starea inițială (prin resetare) microporocesorul realizează accesarea memoriei pentru "fetch", primind codul instrucțiunii simulate – ***NOP***. În continuare, incrementând contorul de adrese, acesta adresează următoarea locație a memoriei - program, primind aceeași instrucțiune vidă. Astfel, pe magistrala de adrese apare periodic toată gama de valori posibile pentru procesorul dat, putându-se verifica funcționarea corectă a decodificatoarelor de adresă și a memoriilor **RAM** și **ROM** din sistem.

Emularea este un proces în cadrul căruia un sistem (de regulă mai puternic, cu mai multe resurse disponibile), denumit ***emulator***, este utilizat pentru imitarea (simularea) funcționării sistemului cu microprocesor supus diagnosticării. Emularea poate fi de mai multe tipuri **[7.]**: "*în fundal*" (care permite doar obținerea modelului logic al sistemului de elaborat prin crearea unui mediu software virtual, fără a reflecta în detalii comportarea în timp real a semnalelor intrărilor și

ieșirilor), emulare "*în prim plan*" (blocurile deja elaborate ale sistemului de emulat sunt realizate în hard și funcționează în timp real, iar blocurile neelaborate se simulează cu ajutorul soft-ului sistemului emulator), sau "*emulare în circuit*" (dirijarea transparentă prin hard a funcționării sistemului din mediul de lucru al emulatorului, prin substituirea microprocesorului emulat fără a modifica parametrii electronici din circuit și fără constrângeri în folosirea oricăror instrucțiuni, spații de memorie, întreruperi sau altor funcții normale ale acestuia).

Generatorul de cuvinte este un aparat predestinat producerii și aplicării de stimuli la intrările sistemului în curs de verificare. Întrucât în sistemele pe bază de microprocesor schimbul de informație se efectuează prin coduri logice cu un număr determinat de biți - cuvinte, aparatul a fost denumit generator de cuvinte. El poate fi utilizat atât pentru diagnosticare, cât și pentru emularea unităților inexistente în cazul proiectării unui sistem nou sau modernizării celui vechi.

În afară de mijloacele specializate, la diagnosticarea echipamentelor cu microprocesor își găsesc aplicare și mijloacele tradiționale utilizate în controlul și depanarea echipamentelor electronicii analogice. De exemplu, ***multimetrul numeric*** se utilizează pentru controlul tensiunii de alimentare, măsurarea curenților consumați de anumite blocuri funcționale ale sistemului, iar ***frecvențmetrul*** – pentru controlul frecvenței semnalului de ceas și măsurarea frecvențelor de pe liniile magistralelor microprocesorului.

Un loc aparte în mulțimea mijloacelor tradiționale de diagnosticare îl ocupă ***osciloscopul***. În cuplaj cu aparatele specializate (de exemplu, cu analizorul logic), el permite deseori depistarea unor defecte a căror sesizare prin alte mijloace ar fi fost foarte dificilă. În ultimul timp a apărut tendința de a extinde posibilitățile osciloscopului până la transformarea lui într-un analizor logic de mare viteză, cu mai multe canale (osciloscoape digitale cu memorare sau module adiționale de tip comutator electronic atașate pentru vizualizarea semnalelor digitale pe mai multe canale).

Pe lângă mijloacele universale tradiționale mai există și o serie de dispozitive simple, manuale, ce se utilizează de mult timp pentru diagnosticarea echipamentelor și sistemelor digitale discrete **[7.]**. Din ele fac parte ***sonda logică analizoare*** (care indică starea logică și prezența impulsurilor într-un singur punct al circuitului), ***sonda logică stimulatoare*** ("pulsatorul", care transferă nodul verificat dintr-o stare logică în alta și îl întoarce în starea inițială prin "injecția" unor impulsuri scurte de curent, fără deconectarea intrării nodului verificat de la ieșirea nodului precedent), ***sonda de curent fără contact*** (care permite depistarea, urmărirea și măsurarea curentului variabil în timp într-un conductor prin inducție electromagnetică, fără contactul electric cu el). Fiind utilizate frecvent în practica diagnosticării datorită accesibilității lor, aceste mijloace devin în unele cazuri mai preferabile chiar decât mijloacele specializate.

3.3. Testarea și punerea la punct a sistemului de dezvoltare.

În cazul sistemului cu **Z80** considerat, în prima parte a punerii la punct am verificat riguros ambele fețe de circuit imprimat ale plăcii prin control vizual și comparație cu desenul original, cât și prin măsurarea cu ohmmetrul a rezistenței traseelor mai subțiri. Cu această ocazie nu au fost depistate erori (controlul vizual a mai fost realizat înaintea corodării plăcii realizate manual, ocazie cu care am depistat unele trasee lipsă și altele incomplete, pe care le-am remediat).

După implantarea componentelor lipite direct pe placă (conectorii, rezistoarele, condensatoarele, diodele și tranzistorii) și a soclurilor pentru circuitele integrate, am efectuat noi controale vizuale ale lipiturilor cu remedierea pe loc a erorilor găsite, am verificat cu ohmmetrul corespondența și absența scurtcircuitelor pe magistralele microprocesorului la toate soclurile circuitelor periferice și memoriilor și am trecut la partea a doua, a punerii sub tensiune.

În prima fază a punerii sub tensiune, am verificat mai întâi prezența și polaritatea corectă a tensiunii de alimentare pe soclurile circuitelor integrate, apoi am montat în socluri numai circuitele **LS 04** și **LS 90**, care constituie generatorul de ceas al sistemului. Prin urmărirea semnalelor pe ecranul unui osciloscop, am verificat startarea corectă și frecvența oscilatorului, funcționarea corectă a divizoarelor cu 2 și cu 5, prezența semnalului de **BAUD**, precum și amplitudinea și forma semnalului de tact **CLK**.

În continuare, am introdus în socluri și cele două cicuite **MAX 232** și am verificat generarea corectă a tensiunilor de **± 10V** pentru conversia **RS 232**, acumulate pe condensatoarele C6 și C7 comune celor două circuite, starea ieșirilor, apoi am "ștrapat" ieșirile la intrări și am verificat dacă tensiunile mari nu pătrund până la intrările circuitului **SIO**, fiind corect convertite **TTL**. La sfârșitul acestei faze, am verificat și comutarea corectă a alimentării **RAM1**, folosind o a doua sursă de alimentare variabilă pe post de "baterie".

În faza a doua a punerii sub tensiune, am montat în soclu microprocesorul **Z80B** (**NMOS**, **6.17 Mhz**) și am conectat pe unul din soclurile circuitelor **LSI** un set de 8 rezistoare de **1K** între pinii magistralei de date și masă. Am verificat astfel resetarea corectă a microprocesorului (atât la conectarea tensiunii de alimentare, cât și la acționarea butonului corespunzător), lipsa aprinderii permanente a LED-ului de pe **/HALT**, și apoi regimul de rulare liberă prin execuția de ***NOP***-uri, cu prezența semnalelor corespunzătoare **/M1**, **/MREQ**, **/RD** și **/REFSH** pe ecranul osciloscopului.

În faza a treia, am conectat în socluri toate circuitele rămase, inclusiv memoriile **RAM** și o memorie **EPROM** cu codul instrucțiunii ***HALT*** în prima locație. După resetarea microprocesorului de la buton, *LED-ul de pe* **/HALT** *nu s-a aprins*, ceea ce indica o funcționare defectuoasă a logicii de selecție a **EPROM**-ului. Totuși, când am introdus în soclu **EPROM**-ul cu programul monitor, sistemul părea să funcționeze corect.

Explicația am găsit-o măsurând ieșirea Q a bistabilului de paginare a memoriei **RAM0** cu **EPROM**-ul, care îndica "1" logic cât timp era apăsat butonul de "RESET", trecând apoi imediat pe "0"logic după eliberarea lui. Practic, primul "fetch" al microprocesorului se făcea din memoria **RAM0**, trecând pe **EPROM** la instrucțiunea următoare, datorită încărcării valorii "0" din bitul 7 al registrului **R** (registrul se încarcă cu valoarea **00H** la resetarea microprocesorului). Așadar, prima instrucțiune din **EPROM** era sărită, în locul ei extrăgându-se un "op-cod" aleator, după cum era încărcată prima locație din **RAM0** la conectarea alimentării.

Am verificat mai întâi schema desenată în **OrCAD**, unde **am descoperit că era conectat semnalul de /RESET pe pinul /PR**ESET **al bistabilului, în loc de pinul /CL**EAR**. Cablajul realizat (din anexele 3. și 4.) prezintă această eroare**, pe care am remediat-o ulterior, prin modificarea traseelor imprimate (într-un mod cât mai puțin evident ...).

Pentru instalarea unui program monitor în memoria **EPROM**, am plecat de la sursa "**NZMON**" a unui monitor **Z80** cu afișaj de tip consolă prin intrefață serială **RS-232** / **SIO**, pe care am modificat-o pentru a fi compatibilă cu resursele hardware ale sistemului proiectat. Principalele modificări le-am făcut în zona secvenței de inițializare a circuitelor periferice, în special a canalului **CTC-1** și a canalului A din circuitul **SIO**, pentru configurarea comunicației seriale la o rată de ***19200 BAUD, 8 biți de date, fără paritate și 1 bit de stop***. Alte modificări sunt realizate în secvența de "shadow" pentru paginarea specifică din sistem și la afișările unor mesaje.

În final, am testat configurarea și funcționarea corectă a tuturor circuitelor periferice din sistem, cu ajutorul programului monitor. Nu am mai descoperit alte erori.

Capitolul 4.

Program de transfer pe port paralel cu PC-ul.

Principalul avantaj al unui sistem de dezvoltare constă în posibilitatea de a încărca programe în memoria sa **RAM** de pe un calculator dotat cu resursele suficiente necesare dezvoltării aplicațiilor software, pentru a le rula și testa astfel funcționalitatea. Programul monitor pe care l-am instalat în memoria **EPROM** avea deja o comandă care permite încărcarea fișierelor de pe un calculator personal **PC** prin interfața serială, în format **INTEL-HEX** (codificare cu caractere **ASCII** în hexazecimal a numerelor binare), dar pentru o versatilitate mai bună este necesară posibilitatea de a încărca în memoria sistemului ***fișiere în format binar***.

Transferul binar al datelor pe un canal serial presupune însă, utilizarea unui protocol destul de sofisticat, cum ar fi protocolul de rețea **XMODEM-CRC**, cu generare/verificare de polinoame cu redundanță ciclică, confirmare și retransmitere în caz de eroare a pachetelor de date. Cu toate că circuitul **SIO** simplifică substanțial implementarea unui astfel de protocol de nivel "data link control" în modul sincron **[2.]**, calculatorul **PC** nu are porturi seriale sincrone. Implementarea unui astfel de protocol pentru modul serial asincron în limbaj de asamblare pe sistemul cu **Z80** este o sarcină destul de dificilă.

Pe de altă parte, un transfer paralel pe 8 biți la distanță scurtă cu linii de "handshake" suplimentare are o viteză mult mai mare și nu necesită implementarea unor protocoale software complicate. Folosirea semnalelor speciale de protocol hardware cu întreruperi în această situație ar permite realizarea unei comunicații de viteză foarte mare, cu un program deosebit de simplu, care ar funcționa și în mediu "multitasking" pentru aplicații în timp real. Totuși, utilizarea întreruperilor nu se justifică aici deoarece, în majoritatea cazurilor când se încarcă un program, sistemul nu mai execută alte sarcini, fiind mai avantajoasă folosirea unui program de transfer cu comanda prin soft a liniilor de protocol.

Un asemenea program de transfer are și avantajul de a putea fi portat pe un sistem **Z80** cu o structură hardware mai simplă, fără interfață serială, circuit "timer" formator de rate, sau chiar fără întreuperi multiple (de exemplu un sistem cu: **Z80-CPU**, **Z80-PIO**, **EPROM**, **RAM** și un singur generator de întrerupere unică cu βE555 , profitând și de **modul 1** de lucru cu întreruperi al microprocesorului, cu salt fix prin ***RST 38H*** **[1.]**).

Din aceste motive am realizat un cablu special de interconectare a circuitului **PIO** din sistem cu portul paralel **LPT** al unui calculator **PC** pentru ***transfer de date binare sub controlul a două programe care rulează pe cele două sisteme de calcul***.

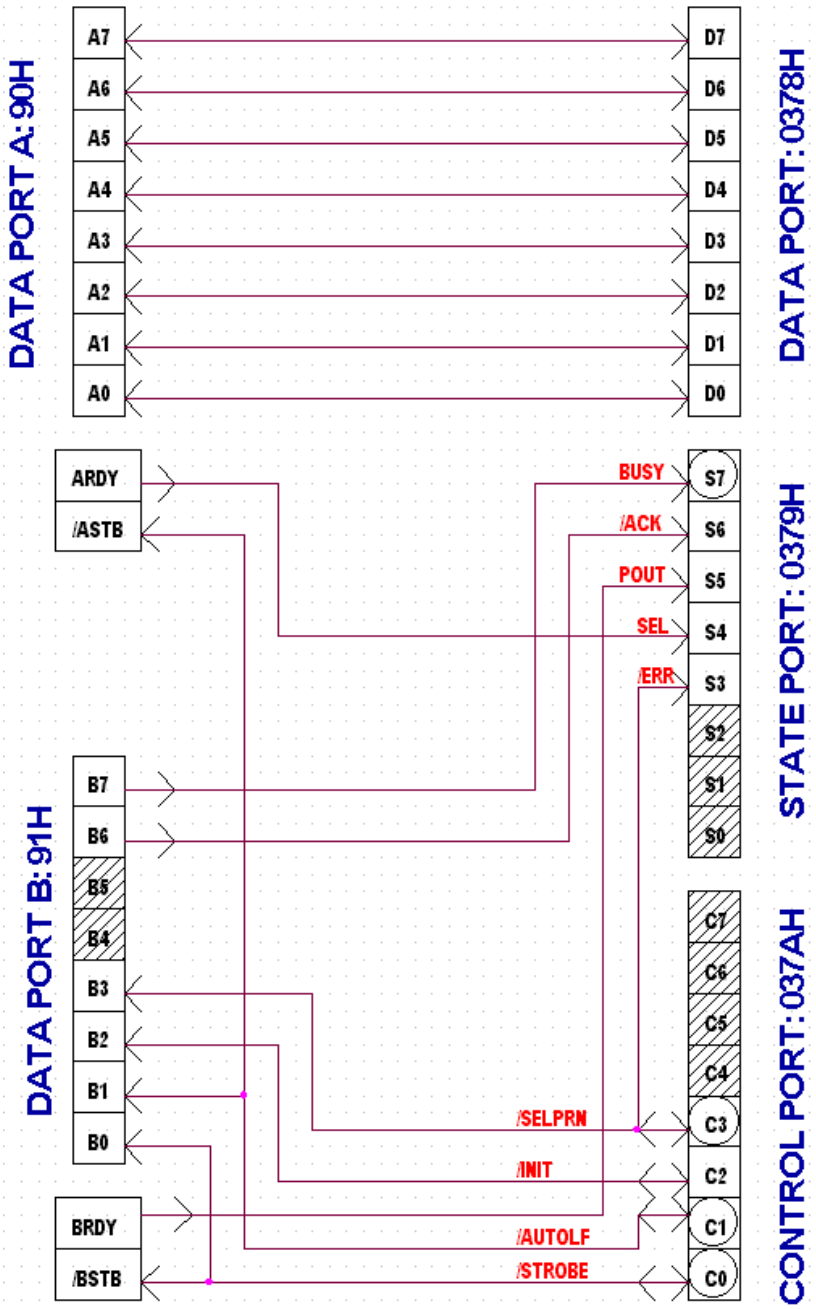


Fig. 4.1.

Cablul de interconectare (**fig. 4.1.**) permite transferul simultan a 8 biți de date de pe orice calculator **PC** pe sistemul cu **Z80** (“download”), pentru transferul invers (“upload”) fiind necesar un calculator **PC** cu port paralel bidirecțional și două programe asemănătoare. În figură am reprezentat schema logică de conectare a porturilor; în partea stângă sunt porturile **A** și **B** ale

circuitului **PIO**, iar în partea dreaptă sunt cele 3 porturi corespunzătoare dintr-un calculator **PC**: ***portul de date*** (ieșire sau bidirecțional, după caz), ***portul de stare*** (intrare) și ***portul de control*** (bidirecțional) **[10.]**. Am încercuit biții din porturile **PC**-ului ale căror linii sunt inversate prin hardware. Biții hașurați sunt nefolosiți (în cazul circuitului **PIO**) sau nu sunt disponibili la conector (fiind rezervați sau utilizați la configurări interne – în cazul **PC**-ului).

Liniile portului **PIO-A** sunt conectate în întregime la portul de date **LPT**, iar portul paralel **PIO-B** este configurat ca intrare pe jumătatea inferioară și ca ieșire pe jumătatea superioară, liniile sale fiind conectate la porturile de stare și control. Am gândit acest mod de legătură în ideea de a utiliza liniile ***B0/B7*** pentru protocolul software cu confirmare la "download" iar liniile ***B1/B6*** pentru protocolul software cu confirmare la "upload".

Conectarea liniei ***C3*** la intrarea ***S3*** oferă posibilitatea de verificare a prezenței cablului la conector, precum și posibilitatea de utilizare a bitului ***B3*** atât în intrare cât și în ieșire (liniile ***B2/B3*** pot fi utilizate la cererea și confirmarea de inversare a sensului de transfer).

Am conectat și liniile de protocol hardware prin întreruperi, pentru a da posibilitatea utilizării și acestui mod de transfer. Ieșirile **ARDY** și **BRDY** sunt conectate la intrările rămase disponibile din portul de stare, iar liniile **/ASTB** și **/BSTB** sunt comandate cu liniile ***C1***, respectiv ***C0***, fiind conectate în paralel cu biții ***B1***, respectiv ***B0***, care rămân nefolosiți în cazul transferului prin întreruperi.

Totodată, acestă conectare permite autogenerarea "strob"-urilor prin program, pentru testarea locală a rutinelor de întrerupere atunci când **PC**-ul nu este conectat (se inversează sensul liniilor ***B1***, ***B0***). Menționez că același cablu l-am folosit și la transferul fișierelor între un calculator **HC-91** cu disc și **PC**, fiind gândit și în acest sens.

Programele de transfer folosesc un ***algoritm de protocol software cu confirmare completă*** (**fig. 4.2.**), pentru a asigura un transfer sigur de informație, chiar dacă vitezele celor două sisteme sunt diferite (viteza de execuție a programului pe **PC** poate varia în limite largi dacă se execută în medii "multitasking"). Protocolul decurge în felul următor:

- sistemul cu **Z80** activează linia **/REQ** (ieșirea ***B7***) semnalând o cerere de date către **PC**, apoi așteaptă trecerea pe zero a liniei de confirmare **/ACK** (intrarea ***B0***);
- calculatorul **PC** detectează trecerea pe zero a liniei **/REQ** (intrarea ***BUSY*** trece pe "1", fiind inversată hardware), plasează octetul pe portul de date și activează linia de confirmare **/ACK** (ieșirea ***/STROBE*** este pusă pe "1", fiind inversată hardware), așteptând în continuare dezactivarea liniei **/REQ**;

– sistemul primește confirmarea **/ACK** (intrarea ***B0*** pe "0" logic), citește datele pe portul **PIO-A**, apoi dezactivează cererea **/REQ**, prin trecerea ieșirii ***B7*** pe "1" logic, așteptând confirmarea încetării cererii de transfer prin trecerea pe "1" a intrării **/ACK**;
– calculatorul **PC** detectează cererea de terminare a transferului prin dezactivarea intrării **/REQ** și timite confirmarea primirii prin dezactivarea ieșirii **/ACK**;

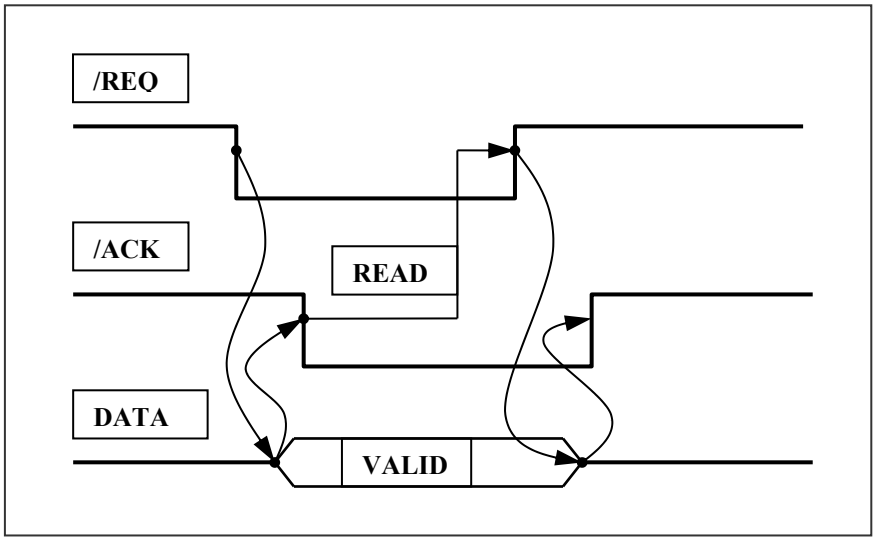


Fig. 4.2.

Cele două sisteme rulează mai întâi o rutină "INITDEVICE", pentru verificarea prezenței corespondentului la capătul celălalt al cablului. Această rutină simulează un ciclu complet de transfer prin activarea liniilor corespunzătoare în același mod, dar cu așteptare limitată în timp a confirmării primirii tranziției semnalelor ("time-out"). Rolul principal al acestei rutine este de a evita blocarea programelor atunci când cablul nu este conectat sau corespondentul nu este pregătit.

De asemenea, ea mai are și funcția de a nu permite începerea transferului dacă semnalele de confirmare sunt găsite deja active, așteptând nu o anumită valoare a semnalului pe liniile de confirmare, ci un răspuns dinamic la modificările generate pe liniile de cerere. Acest lucru permite ca ***cele două sisteme comunicante să se aștepte reciproc un anumit timp (30 de secunde), indiferent pe care a fost pornit prima dată programul de comunicație***.

Calculatorul **PC** transmite mai întâi adresa pe 2 octeți unde se va încărca programul în memoria sistemului, iar apoi transmite și lungimea datelor, în număr (codificat tot pe 16 biți) de sectoare – buffer (pachete de date cu lungimea de ***128 octeți***), calculat automat la deschiderea fișierului de transmis de pe disc. Se permite astfel transferarea unor fișere mai mari decât ***64Ko*** (16 biți), de exemplu pe discul calculatorului **HC-91** (am folosit aceste primitive și la realizarea unor programe de transfer în acest scop).

Adresa și lungimea datelor transferate sunt transmise de două ori consecutiv de către **PC** și verificate de către sistem la începutul transferului, pentru evitarea încărcării eronate (adresa greșită) sau blocării transferului (lungimea greșită), printr-o eventuală pierdere a primului octet transmis.

Programul executat pe sistemul de dezvoltare mai are și o funcție suplimentară de semnalare a stării în care se găsește transferul, prin realizarea unor "efecte" pe bitul **/DTRB**, care pot fi ascultate cu ajutorul unui mic difuzor piezoceramic sau urmărite cu osciloscopul. În perioada de așteptare a stabilirii legăturii subrutina "INIDEV" creează câte un impuls la fiecare decrementare a contorului de așteptare, iar în timpul transferului propriu-zis subrutina "PIORD" activează linia **/DTRB** la începutul unui bloc de date de 128 de octeți și o dezactivează după ce acesta a fost transferat.

Dacă a expirat timpul de așteptare, subrutina "INIDEV" iese cu "flag"-ul **Z** (zero) setat și lasă linia **/DTRB** activată, pentru o eventuală depistare a "time-out"-ului (altfel, iese cu **Z** = "0" și inactivează **/DTRB**). Subrutina "PIORD" este și optimizată pentru obținerea unei viteze cât mai mari de transfer a blocurilor de date (prin utilizarea instrucțiunii ***INI*** și a registrului secundar **AF'**).

Programul, prezentat în continuare, este semnificativ ca exemplu de utilizare a setului de instrucțiuni și posibilităților microprocesorului **Z80** și **modului 3** al circuitului **PIO** (la citirea portului de protocol se citește și starea ieșirilor, nemaifiind necesară memorarea lor într-un "octet - martor").

```
ADAT      EQU  90H        ; adresa portului de date PIO-A;
BDAT      EQU  91H        ; adresa portului de date PIO-B;
ACOM      EQU  92H        ; adresa portului de comenzi PIO-A;
BCOM      EQU  93H        ; adresa portului de comenzi PIO-B;
DTRB      EQU  8BH        ; adresa portului de control/stare SIO-B;

PIODLD:   CALL INIPIO     ; inițializează circuitul PIO pentru transfer;
          LD   DE,10000
          CALL INIDEV     ; "time-out" după 30 de secunde;
          RET  Z
          LD   B,4
          LD   HL,ADR
          CALL PIORD      ; încarcă variabilele de 16 biți ADR și LEN;
          LD   HL,(ADR)
          LD   DE,(LEN)
          EXX             ; păstrează adresa și lungimea în regiștrii secundari;
          LD   B,4
          LD   HL,ADR
          CALL PIORD      ; mai încarcă o dată 4 locații la adrese consecutive;
          EXX             ; readuce vechile valori în lucru;
```

```
            LD    BC,(ADR)
            OR    A           ; șterge "Carry" pentru scădere;
            SBC   HL,BC
            ADD   HL,BC       ; comparație pe 16 biți fără modificarea registrelor;
            JR    NZ,LDERR    ; iese dacă adresele nu coincid;
            EX    DE,HL       ; acum compară lungimile;
            LD    BC,(LEN)
            OR    A
            SBC   HL,BC       ; nu se mai reface registrul HL;
            JR    NZ,LDERR    ; iese dacă lungimile nu coincid;
            EX    DE,HL       ; acum HL conține adresa și BC lungimea;
LDBLK:      PUSH  BC
            LD    B,128       ; încarcă 128 de octeți la adrese succesive;
            CALL  PIORD       ; folosind HL ca pointer și iese cu HL+128;
            POP   BC
            DEC   BC          ; decrementează numărul de blocuri (lungimea);
            LD    A,B         ; compară BC cu zero;
            OR    C
            JR    NZ,LDBLK    ; continuă dacă contorul nu a ajuns la zero;
LDERR:      CALL  RESPIO      ; aduce circuitul PIO în starea inițială;
            RET

; rutina INIDEV validează transferul așteptând un timp: DE × 3 ms;
; conectarea partenerului de comunicație;

INIDEV:     IN    A,(BDAT)
            XOR   80H
            OUT   (BDAT),A    ; complementează bitul B7, activând linia /REQ;
            LD    H,D         ; păstrează contorul de timp și în HL;
            LD    L,E
DVSTB0:     CALL  EFCTON      ; activează linia /DTRB;
            DEC   DE
            LD    A,D
            OR    E
            RET   Z           ; iese dacă a expirat numărul de iterații;
            CALL  EFCTOF      ; produce efecte sonore pe /DTRB;
            LD    BC,150
            CALL  TEMP        ; așteaptă 3 milisecunde activarea /ACK;
            IN    A,(BDAT)
            BIT   0,A
            JR    NZ,DVSTB0   ; buclează dacă linia B0 nu a trecut pe zero;
            XOR   80H
            OUT   (BDAT),A    ; dezactivează /REQ;
DVSTB1:     CALL  EFCTON
            DEC   HL          ; decrementează contorul de iterații inițial;
            LD    A,H
            OR    L
            RET   Z           ; "time-out" cu Z=1 și /DTRB activ;
            CALL  EFCTOF
            LD    BC,150
```

```
            CALL TEMP
            IN   A,(BDAT)
            BIT  0,A
            JR   Z,DVSTB1   ; buclează până /ACK este inactivat;
            RET

; rutina PIORD încarcă în memorie un număr de octeți dat în BC, folosind ca pointer
; registrul HL, care este returnat modificat, indicând adresa următoare ultimei locații;

PIORD:      LD   DE,8001H   ; măștile pentru biții B7 și B0 în D și E;
            LD   C,ADAT     ; portul de date pentru intrare indirectă cu INI;
            CALL EFCTON
            DI              ; dezactivează întreruperile la transferul blocului;
LDRDY:      EX   AF,AF'
            IN   A,(BDAT)
            XOR  D
            OUT  (BDAT),A   ; activează B0 prin complementare;
            EX   AF,AF'     ; și păstrează imaginea ieșirilor în AF';
RDSTB0:     IN   A,(BDAT)
            AND  E
            JR   NZ,RDSTB0  ; așteaptă până B0 devine zero;
            INI             ; port(C)=>(HL), INC HL, DEC B, Z=1 dacă B=0;
            EX   AF,AF'     ; aduce imaginea portului PIO-B și păstrează Z;
            XOR  D
            OUT  (BDAT),A   ; dezactivează cererea prin complementare;
RDSTB1:     IN   A,(BDAT)
            AND  E
            JR   Z,RDSTB1   ; așteaptă până linia /ACK trece și ea pe "1";
            EX   AF,AF'     ; aduce "flag"-ul Z corespunzător testului: B=0 ?;
            JR   NZ,LDRDY
            CALL EFCTOF
            EI              ; activează întreruperile aici pt. a nu încărca stiva;
            RET             ; Z80 validează EI după RET;

INIPIO:     DI
            LD   A,0CFH     ; PIO-A în modul 3;
            OUT  (ACOM),A
            LD   A,0FFH     ; toate liniile portului PIO-A sunt intrări;
            OUT  (ACOM),A
            LD   A,0CFH     ; PIO-B în modul 3;
            OUT  (BCOM),A
            LD   A,0FH      ; B7-B4 sunt ieșiri și B3-B0 intrări;
            OUT  (BCOM),A
            LD   A,0F0H
            OUT  (BDAT),A   ; ieșirile sunt inițial inactive (în "1" logic);
            EI
            RET

RESPIO:     DI
            LD   A,0CFH
```

```
            OUT   (ACOM),A   ; PIO-A în modul 3;
            LD    A,0FFH
            OUT   (ACOM),A   ; toate liniile portului sunt intrări;;
            LD    A,0CFH
            OUT   (BCOM),A   ; PIO-B în modul 3;
            LD    A,0FFH
            OUT   (BCOM),A   ; toate liniile PIO-B sunt intrări;
            EI
            RET

EFCTON:     LD    A,05H
            OUT   (DTRB),A   ; setează mai întâi registrul WR5 care ține DTR;
            LD    A,0E0H
            OUT   (DTRB),A   ; bit7=DTR=1, bit1=RTS=0, nu strică transmisia;
            RET

EFCTOF:     LD    A,05H
            OUT   (DTRB),A   ; selectează registrul de control WR5 din SIO-B;
            LD    A,60H
            OUT   (DTRB),A   ; modifică înapoi DTR=0, restul biților - nu;
            RET

TEMP:       DI               ; întârziere 122T × BC ori;
            EX    (SP),IX    ; interschimbă de 2 ori vârful stivei cu registrul IX;
            EX    (SP),IX    ; 2 × 23T, pentru pierderea timpului ...;
            EX    (SP),IY    ; acum pierde timpul cu IY;
            EX    (SP),IY    ; 2 × 23T, fără alt efect nedorit ...;
            DEC   BC
            LD    A,B
            OR    C
            JR    NZ,TEMP    ; repetă 122T până când BC a ajuns la zero;
            EI
            RET

; variabilele în care se încarcă la început adresa și lungimea;
; trebuie să fie în RAM, la adrese succesive;

ADR:        DEFW 8000H       ; aici se memorează adresa de "download";
LEN:        DEFW 0000H       ; imediat după se ține lungimea inițială;
```

Programul care rulează pe **PC** permite citirea unui fișier binar de pe disc și transferul său la adresa dorită din memoria microsistemului cu **Z80**. Buclele de protocol software au fost realizate mai întâi în "**C**", dar execuția lor era deosebit de lentă (mai lentă decât execuția programului în asamblare de pe sistemul de dezvoltare), asa că le-am rescris în limbaj de asamblare **X86**. "Listing"-ul comentat al programului este prezentat în continuare:

```
#include <stdio.h>
#include <conio.h>
#include <io.h>
#include <dos.h>
#include <stdlib.h>
#include <fcntl.h>
#define LPT 0X0378 // adresa unde sunt mapate cele 3 porturi succesive;

void initdevice(void)
     {
          asm MOV CX,30000      ; se fac 30000 de buclări pentru fiecare nivel /REQ;
          asm PUSH CX           ; păstrează contorul de iterații CX;
strob1:   asm DEC CX
          asm JCXZ timeout      ; iese dacă contorul decrementat a ajuns la zero;
          asm PUSH CX
          delay(1);             // întârziere 1 ms fără a afecta CX;
          asm POP CX
          asm MOV DX,LPT+1
          asm IN AL,DX          ; citește portul de stare;
          asm AND AL,0X80
          asm JZ strob1         ; așteaptă în buclă activarea (/REQ=0)=>(S7=1);
          asm POP CX            ; reface contorul pt. a doua buclă de așteptare;
          asm MOV DX,LPT+2
          asm IN AL,DX          ; citește portul de control;
          asm XOR AL,0X01
          asm OUT DX,AL         ; activează confirmarea (C0=1)=>(/ACK=0);
          asm PUSH CX
strob0:   asm DEC CX
          asm JCXZ timeout
          asm PUSH CX
          delay(1);
          asm POP CX
          asm MOV DX,LPT+1
          asm IN AL,DX          ; citește portul de stare;
          asm AND AL,0x80
          asm JNZ strob0        ; și așteaptă în bucla contorizată până S7=0;
          asm POP CX            ; descarcă stiva;
          asm MOV DX,LPT+2
          asm IN AL,DX          ; citește portul de control;
          asm XOR AL,0X01
          asm OUT DX,AL         ; și dezactivează linia /ACK prin complementare;
          asm RET               ; întârziere 1 ms fără a afecta CX;
timeout:  asm POP CX            ; descarcă stiva;

          printf("\nTIME OUT: device not responding!\n");
          exit(0);
     }

void send(unsigned char data)
     {
```

```
          asm MOV DX,LPT+1
ready1:   asm IN AL,DX          ; citește portul de stare;
          asm AND AL,0X80
          asm JZ ready1         ; și buclează până la apariția unei cerei /REQ;
          asm MOV DX,LPT
          asm MOV AL,data
          asm OUT DX,AL         ; plasează octetul "data" pe portul de date;
          asm MOV DX,LPT+2
          asm IN AL,DX
          asm XOR AL,0X01
          asm OUT DX,AL         ; confirmă prezența datelor cu /ACK;
          asm MOV DX,LPT+1
ready0:   asm IN AL,DX          ; citește portul de stare;
          asm AND AL,0x80
          asm JNZ ready0        ; și așteaptă în buclă terminarea citirii octetului;
          asm MOV DX,LPT+2
          asm IN AL,DX
          asm XOR AL,0X01
          asm OUT DX,AL         ; confirmă terminarea recepției;
     }

void lptwrite(void)
     {
     char name[12];
     int handle,nbytes;
     unsigned char pbuf[128];
             // datele de pe disc se citesc într-un buffer de 128 de octeți;

     unsigned int i,j,addr,size;
     unsigned long int len; // lungimea fișierului poate fi pe 4 octeți;


     outportb(LPT+2,0x00);   // pune ieșirea C0 (inversată hardware) pe "1" logic;
     outportb(LPT,0x00);     // inițializează portul de date;

     do {     // se așteaptă introducerea unui nume de fișier (valid în directorul curent);

        printf("Input file name: "); scanf("%s",name);
        handle = open(name,O_RDONLY|O_BINARY);
             // deschide fișierul numai pentru citire în acces binar de tip DOS;

        if (handle==-1) printf("\nFile not found!\n");
        } while (handle==-1);

     len = filelength(handle); size = (unsigned int)(1+len/128);
             // lungimea se măsoară în blocuri de 128 de octeți;

     printf("File size: %lu Kbytes = %lu bytes\n", len>>10,len);
     printf("         = %u blocks of 128 bytes\n",size);
```

```
    printf("\nDownload address (in hex): "); scanf("%X",&addr);

    printf("Waiting for response...\n");
    initdevice();
    printf("Transfering data at %XH...\n",addr);

    for (i=0;i<2;i++) {
            // transmite de două ori adresa de încărcare și lungimea pe 16 biți a datelor;

                    send((unsigned char)(addr));
                    send((unsigned char)(addr>>8));
                    send((unsigned char)(size));
                    send((unsigned char)(size>>8));
                    }

    for (i=0;i<size;i++)
        {
        nbytes = read(handle,pbuf,128); // citește "nbytes" în buffer;

        if (nbytes==128) for (j=0;j<128;j++) send(pbuf[j]);
            // trimite tot blocul citit în buffer dacă are exact 128 de octeți;

        else {
             for (j=0;j<nbytes;j++) send(pbuf[j]);
                    // altfel trimite mai întâi câți octeți s-au citit;

             for (j=nbytes;j<128;j++) send(0x00);
                    // și completează cu zero până se fac 128 de octeți;
             }
        printf("%u-blocks\r",i+1);
        }

    close(handle); // închide fișierul;
    printf("\nLPT transfer complete.\n");
    outportb(LPT+2,0x00);   // pune ieșirea C0 în starea inactivă din nou;
    outportb(LPT,0x00);     // resetează portul de date pe zero înainte de ieșire;
    }


void main()
{
clrscr();
printf("-------------------------------------\n");
printf("PC-X86  LPT-parallel port downloader\n");
printf("LPT: I/O=sb7/cb0 - software protocol\n");
printf("Soft By ALEO,    copyright 2000,2003\n");
printf("-------------------------------------\n");

lptwrite();

}
```

Capitolul 5.

Concluzii.

Deși aparent "învechit", sistemul de dezvoltare cu **Z80** poate oferi multe satisfacții și posibilități de lucru utilizatorilor, prin felxibilitatea sa, prin puterea de calcul și a resurselor hardware suficiente disponibile și nu în ultimul rând prin evidențierea detaliilor intime de proiectare, care au un caracter sugestiv pentru acest tip de sisteme de calcul. În acest sens se poate spune că un astfel de sistem poate "umple golul" existent între montajele cu microcontrollere și sistemele mari de calcul, de tip calculator personal – **PC**.

Este interesant de remarcat combinația de circuite clasice cu o serie de circuite moderne, sistemul fiind proiectat într-un context actual (prin folosirea memoriilor **RAM** statice de **32Ko** în locul unor memorii dinamice de asemenea capacități specifice sistemelor clasice cu **Z80**, prin utilizarea circuitelor de conversie serială **MAX 232** cu autoalimentare în locul convertoarelor de nivel **1488** și **1489** și în primul rând prin prezența controllerului **PCF 8584** pentru magistrala modernă – **I2C** și a extensiei pentru **LCD**).

Gama de aplicațiii unde sistemul își poate dovedi cel mai bine utilitatea este în domeniul conducerii automate a proceselor tehnice, drept calculator de proces care poate controla un echipament suficient de complex pentru a nu putea fi condus cu microcontroller sau mai multe echipamente simple care se folosesc concurent în procesul de producție. Posibilitățile mari de extensie și de interfațare cu diverse echipamente și dispozitive industriale, fac ca prezența unui astfel de sistem ca un microcalculator autonom între procesul tehnic condus și un calculator **PC** cu interfață grafică pentru operatorul uman să fie extrem de folositoare.

Protocolul și programele de comunicație prezentate pot fi generalizate și translatate în scopul utilizării lor la transferul paralel de date între diverse dispozitive și echipamente de calcul, având un caracter suficient de universal și o funcționalitate destul de bună.

Deoarece nu am dorit neapărat proiectarea și realizarea unui echipament cu succes comercial, ci demonstrarea anumitor cunoștințe acumulate în lucrul cu familia microprocesorului **Z80** și evidențierea anumitor facilități deosebite ale acestor circuite care au început să fie pierdute din vedere, consider că mi-am atins scopul propus.

Bibliografie

[1.] Patrubány M. – *Totul despre microprocesorul Z80*, vol. 1 și 2, Ed. Tehnică, București, 1989;

[2.] Lupu C., Stăncescu S. – *Microprocesoare. Circuite. Proiectare.*, Ed. Militară, București, 1986;

[3.] Sztojanov I., Borcoci E., Tomescu N., Bulik D., Petrec M., Petrec C. – *De la poarta TTL la microprocesor*, vol. 1 și 2, Ed. Tehnică, București, 1987;

[4.] Mureșan T., Strugaru C., Stoinescu R., Petriu E. – *Microprocesorul 8080 în aplicații*, Ed. Facla, Timișoara, 1981;

[5.] Pleter O., Constantinescu C. – *Z80 în microcalculatoare compatibile Spectrum*, Ed. Militară, București, 1995;

[6.] Huțanu C., Postolache M. – *Sisteme cu microprocesoare în conducerea automată a proceselor*, vol. 1, ediția a 2-a, Ed. Academica, Iași, 2001;

[7.] Anatol G., Pintilie D. – *Diagnosticarea tehnică a echipamentelor microprocesor*, Ed. Universitas, Chișinău, 1992;

[8.] *** *Dispozitive și circuite electronice I* – note de curs, 1999;

[9.] *** *Circuite numerice* – note de curs, 2000;

[10.] *** *Calculatoare numerice I, II* – note de curs, 2000-2001;

[11.] *** *Proiectarea cu microprocesoare I, II* – note de curs, 2001-2002;

[12.] *** *Microcalculatorul TIM-S. Manual de funcționare și scheme logice* – ITCI-FMECTC, Timișoara, 1986;

[13.] *** *Microcalculatorul COBRA. Manual de funcționare și scheme logice* – ITCI Brașov, 1992;

[14.] *** *Full Line Condensed Catalog* – IPRS Băneasa, 1993-1994;

[15.] *** *High Speed CMOS Logic Databook* – Texas Instruments, 1993;

[16.] *** *Z80 Family CPU & Peripherals User's Manual* – Zilog, Inc., 2000-2001;

[17.] *** *Z80 Microprocessor Family Databook* – SGS-ATES, 3rd Edition, 1982;

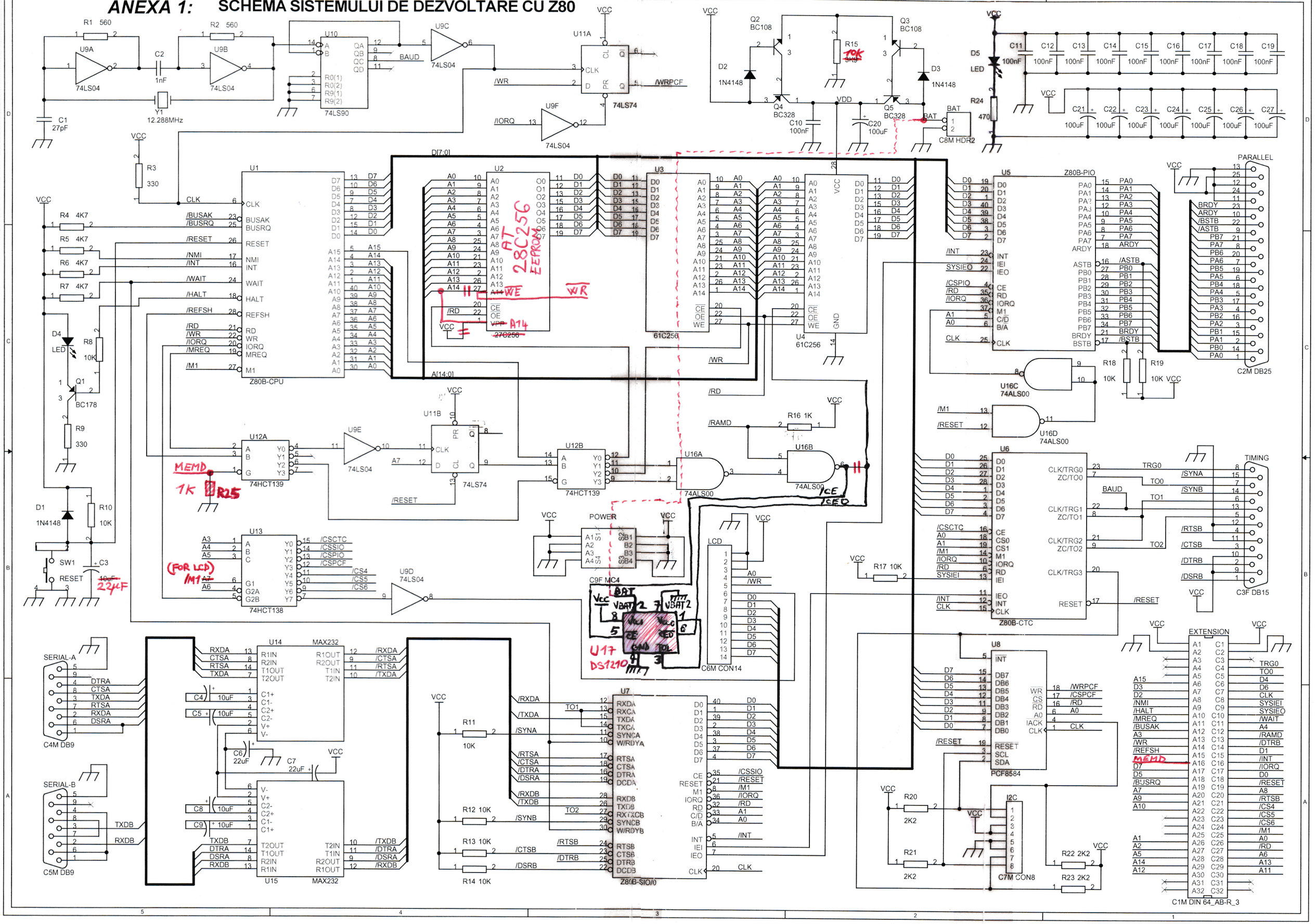
[18.] *** *PCF8584, I2C-bus contoller data sheet* – Philips Semiconductors, 1997;

[19.] *** *OrCAD Capture User's Guide, OrCAD Layout User's Guide* – Cadence Design Systems Inc., Second Edition, 2000;

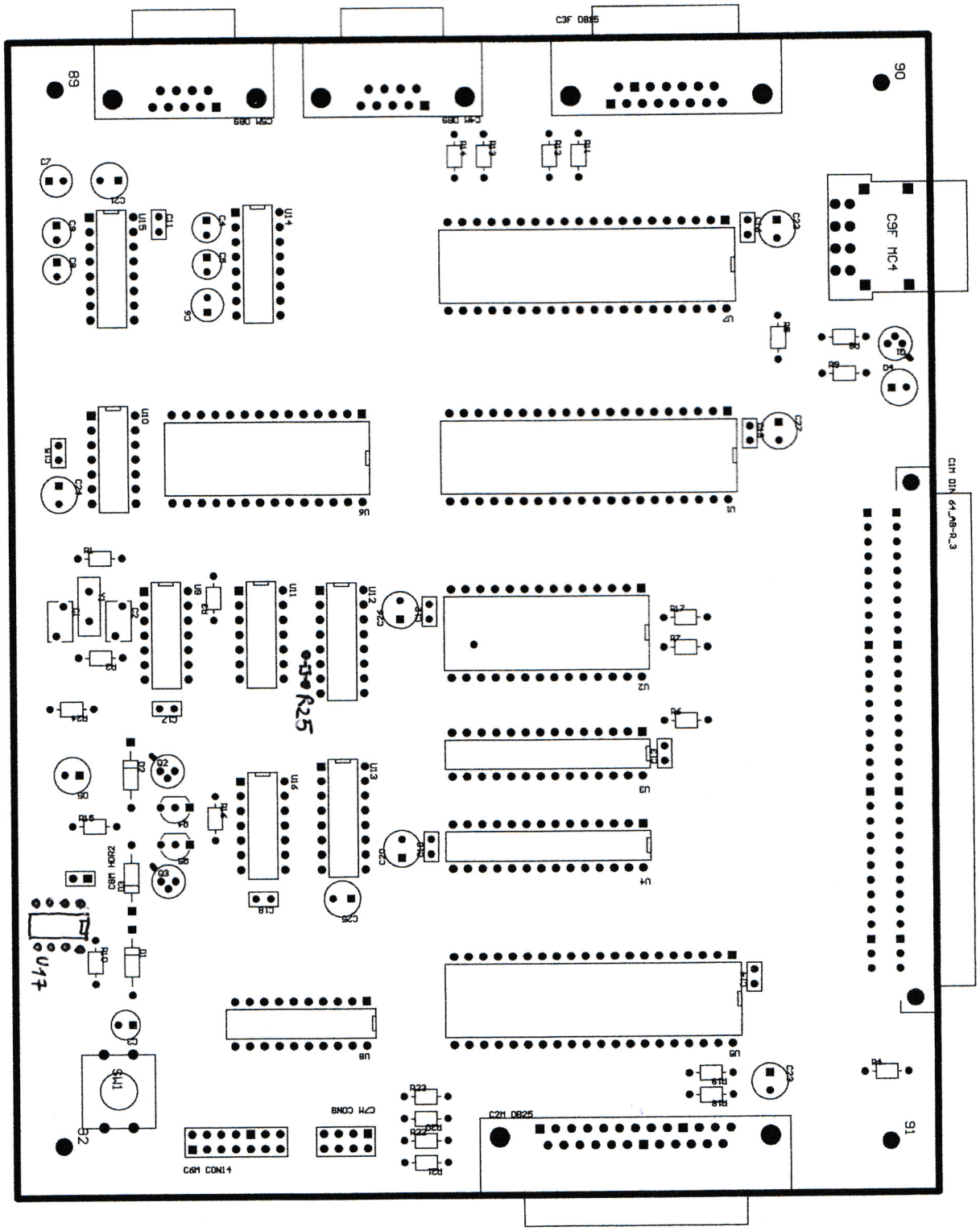
[20.] *** *Z80 CPU Official Support Page* – pagină WEB, "http://www.geocities.com/SiliconValley/Peaks/3938/z80_home.htm", 2000;

[21.] *** *ICL 232, +5V Powered Dual RS-232 Transmitter / Receiver data sheet* – Harris Semiconductor, 1993;

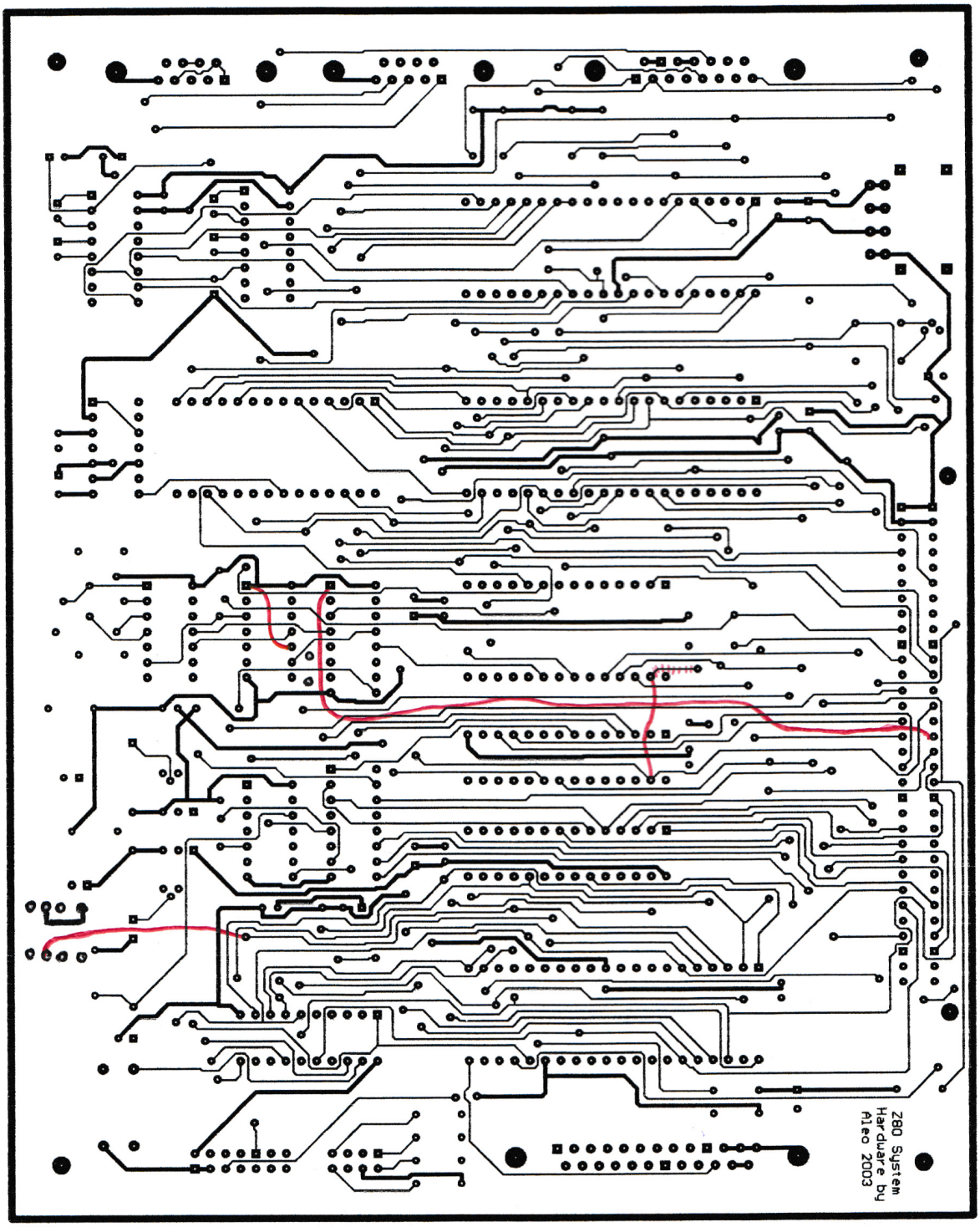
ANEXA 1: SCHEMA SISTEMULUI DE DEZVOLTARE CU Z80



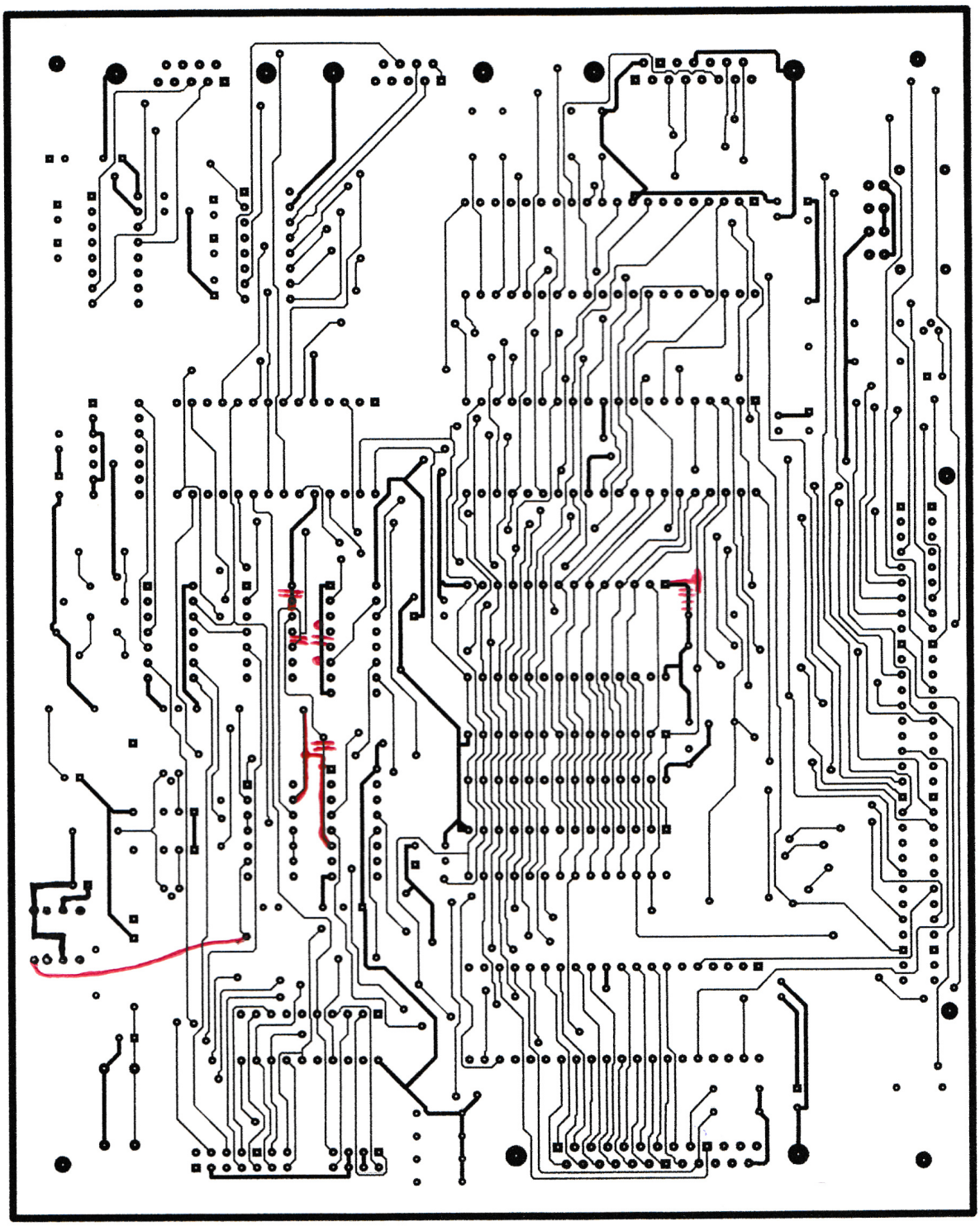
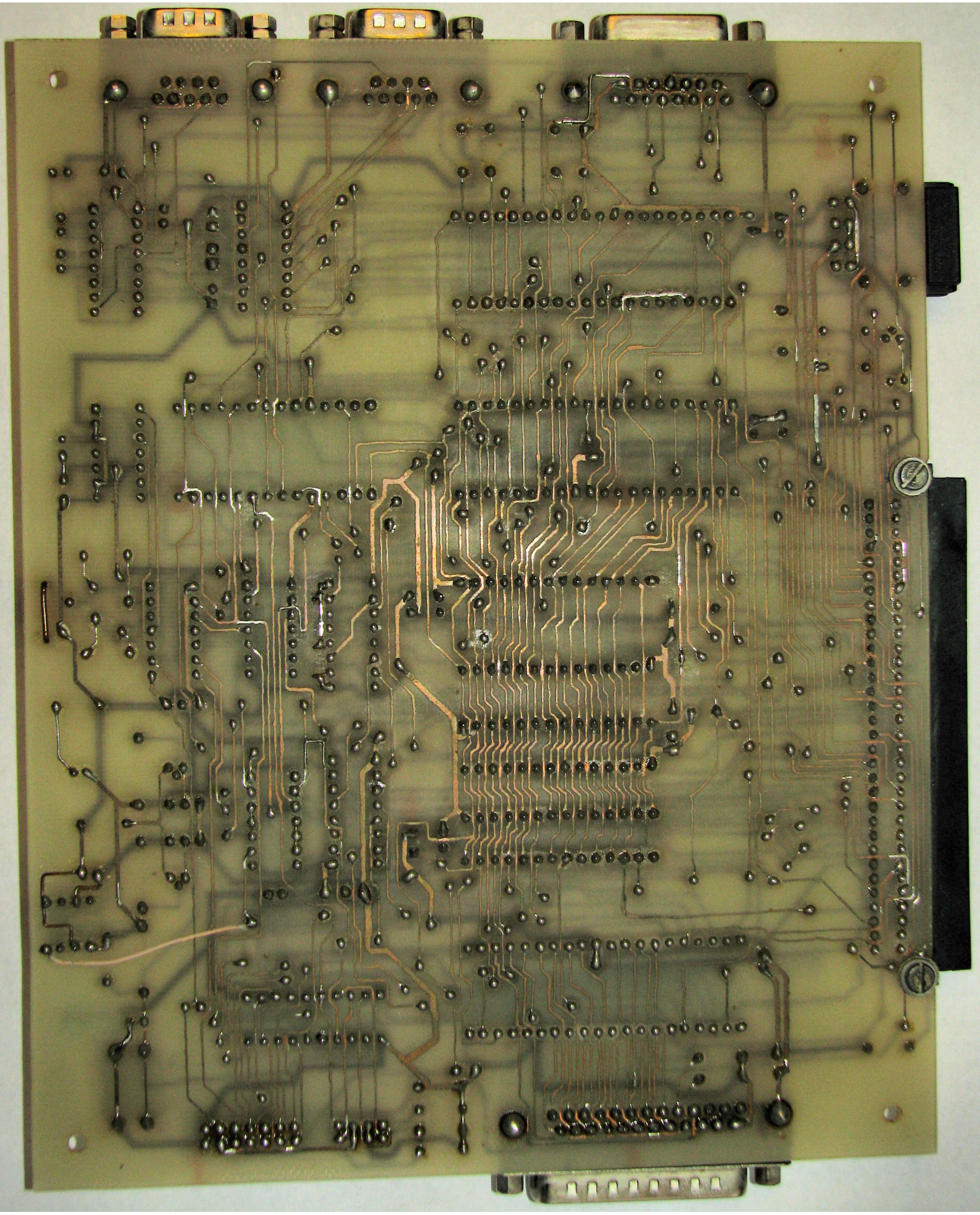
Anexa 2.


C3F DB15
C3M DB9
C2M DB9
C3F MC4
C1M DIN 64_AB-R_3
C2M DB25
C6M CON14
C7M CON8
R25
U17

Anexa 3.



Anexa 4.



Z84C0088CV
Z80BCPU
28842
Z84C3006PEC
Z80 CTC
9804
PM
Z84C0088CV
Z80BSIO-0
28609
Z84C20B PS
Z80B PIO
8735
UM61256AK-15
9524S
N42734
UM61256AK-15
9524S
N42734
AT29C256
70PL
0220
HD74LS04P
6F28
MH74LS74
PCF8582P
TK8774.1
KMM01252
DS1210
0144C1
007AA
SGS
2SD02A
3S-87